



**UNIVERSIDADE FEDERAL DO PARÁ  
CAMPUS UNIVERSITÁRIO DE TUCURUÍ  
FACULDADE DE ENGENHARIA ELÉTRICA**

**OSÉIAS DIAS DE FARIAS**

**AEROPÊNDULO: IMPLEMENTAÇÃO DE UM LABORATÓRIO VIRTUAL PARA  
ESTUDOS DE MODELAGEM E CONTROLE DE SISTEMAS DINÂMICOS**

**TUCURUÍ-PA  
2023**



UNIVERSIDADE FEDERAL DO PARÁ  
CAMPUS UNIVERSITÁRIO DE TUCURUÍ  
FACULDADE DE ENGENHARIA ELÉTRICA

OSÉIAS DIAS DE FARIAS

**AEROPÊNDULO: IMPLEMENTAÇÃO DE UM LABORATÓRIO VIRTUAL PARA  
ESTUDOS DE MODELAGEM E CONTROLE DE SISTEMAS DINÂMICOS**

Trabalho de conclusão de curso apresentado ao colegiado da Faculdade de Engenharia Elétrica, do Campus Universitário de Tucuruí, da Universidade Federal do Pará, como requisito necessário para obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Raphael Barros Teixeira

TUCURUÍ-PA

2023

OSÉIAS DIAS DE FARIAS

**AEROPÊNDULO: IMPLEMENTAÇÃO DE UM LABORATÓRIO VIRTUAL PARA  
ESTUDOS DE MODELAGEM E CONTROLE DE SISTEMAS DINÂMICOS**

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À BANCA EXAMINADORA APROVADA PELO COLEGIADO DA FACULDADE DE ENGENHARIA ELÉTRICA.

DATA DE APROVAÇÃO: XX / XX / XXXX

CONCEITO:

BANCA EXAMINADORA:

---

Prof. Dr. Raphael Barros Teixeira  
Orientador / UFPA-CAMTUC-FEE

---

Prof. Dr. Rafael Suzuki Bayma  
Membro / UFPA-CAMTUC-FEE

---

Prof. Dr. André Felipe Souza da Cruz  
Membro / UFPA-CAMTUC-FEE

TUCURUÍ-PA

2023

Dedico a minha mãe, meu pai e a minhas irmãs,  
além de todas as pessoas que me apoaram nessa  
fase crucial da minha vida.

# **AGRADECIMENTOS**

---

---

À minha mãe e meu pai que dedicaram suas vidas por esse propósito;  
Aos professores do curso pela dedicação em simplificar o complexo, em especial aos professores da área de sistema de controle;  
Finalmente, ao meu orientador Prof. Dr. Raphael Teixeira pela sua orientação e paciência.

*“Demore o tempo que for para decidir o que você quer da vida, e depois que decidir não recue ante nenhum pretexto, porque o mundo tentará te dissuadir.”*  
*(Friedrich Wilhelm Nietzsche )*

# Resumo

Este trabalho apresenta um laboratório virtual abrangente para o estudo de sistemas de controle, que combina a integração de um protótipo físico, simulador 3D e uma interface gráfica interativa. A motivação para este projeto reside na intrínseca complexidade associada à compreensão de sistemas de controle, que frequentemente apresenta desafios, sobretudo para estudantes que precisam transpor a barreira de abstrair sistemas físicos em termos de equações matemáticas. Para o desenvolvimento do projeto foi implementado um protótipo do Aeropêndulo completo com um conjunto de software que permite a interação do usuário com o sistema físico, possibilitando ao usuário realizar modificações nos parâmetros do sistema em tempo real, além disso, foi elaborado um gêmeo digital para espelhar a dinâmica do protótipo do Aeropêndulo a partir de um simulador 3D, por fim, foi realizado testes para a validação do laboratório, sendo eles: aplicação de identificação de sistema usando função de transferência discreta e mínimos quadrados e teste em malha fechada com controlador PID. O projeto foi hospedado no GitHub, a fim de disseminar o conhecimento e permitir que entusiastas, estudantes e pesquisadores tenham acesso ao projeto completo, A combinação de protótipos, simuladores, interface gráfica e integração de diversas disciplinas da Engenharia Elétrica cria um ambiente de aprendizado envolvente e eficaz para o estudo de sistemas de controle. Essa abordagem reduz a resistência inicial dos alunos e promove uma compreensão mais profunda dos conceitos e aplicações dessa área. Por meio da prática, os estudantes podem perceber a relevância das abstrações matemáticas na resolução de problemas reais, preparando-os para lidar com sistemas de controle complexos e desafiadores.

**Palavras Chave:** Aeropêndulo, identificação de sistema, protótipo, simulador, programação.

# Abstract

Texto do abstract (inglês)

**Keywords:** Palavras chave em inglês.

# LISTA DE FIGURAS

---

---

Figura 1 – Diagrama esquemático do Aeropêndulo.	6
Figura 2 – Subsistemas do Aeropêndulo.	7
Figura 3 – Diagrama esquemático do Braço do Aeropêndulo.	8
Figura 4 – Motor CC Série.	10
Figura 5 – Diagrama Elétrico/Mecânico Motor CC Série.	11
Figura 6 – Diagrama da junção dos subsistemas do Aeropêndulo.	16
Figura 7 – Protótipo do Aeropêndulo.	17
Figura 8 – Chapas de compensado.	18
Figura 9 – Tubo de Fibra de Vidro de Carbono 3x3x2mm.	18
Figura 10 – Potenciômetro $50k\Omega$ .	19
Figura 11 – Placa de desenvolvimento Esp32.	20
Figura 12 – Fonte Chaveada 2A, 5V, 25W.	20
Figura 13 – Módulo Driver L298n.	21
Figura 14 – Conjunto (suporte motor cw/ccw hélice).	22
Figura 15 – Componentes eletrônicos.	22
Figura 16 – Diagrama de comunicação do Aeropêndulo.	23
Figura 17 – Esquema de conexões elétricas do Aeropêndulo.	24
Figura 18 – Gêmeo Digital - Simulador com VPython.	26
Figura 19 – Diagrama da arquitetura do Gêmeo Digital.	27
Figura 20 – Interface Gráfica.	31
Figura 21 – Partes da Interface Gráfica.	32
Figura 22 – Arquitetura do Firmware do Aeropêndulo.	35
Figura 23 – Diagrama do Laboratório Virtual.	36
Figura 24 – Sinal PRBS de entrada.	38
Figura 25 – Sinal PRBS de saída.	38
Figura 26 – Dados de identificação e validação do modelo.	40
Figura 27 – Validação do modelo de segundo grau.	42
Figura 28 – Validação do modelo de segundo grau.	44
Figura 29 – Sistema em Malha Fechada com Controlador PID.	45
Figura 30 – Gráficos dos Estados do Aeropêndulo com Controlador PID.	46
Figura 31 – Gêmeo Digital com Controlador PID.	46
Figura 32 – Sistema em Malha Fechada com Controlador PID.	47

Figura 33 – Gêmeo Digital consumindo os dados em tempo real do protótipo em Malha Fechada com Controlador PID . . . . .	47
Figura 34 – Repositório no GitHub. . . . .	50

# SUMÁRIO

---

---

<b>Resumo</b>	<b>vii</b>	
<b>Abstract</b>	<b>viii</b>	
<b>Lista de figuras</b>	<b>ix</b>	
<b>Sumário</b>	<b>xi</b>	
<b>1</b>	<b>INTRODUÇÃO</b>	<b>1</b>
<b>1.1</b>	<b>Justificativa</b>	<b>1</b>
<b>1.2</b>	<b>Objetivos</b>	<b>3</b>
<b>1.2.1</b>	Objetivos Gerais	3
<b>1.2.2</b>	Objetivos Específicos	3
<b>1.3</b>	<b>Escopo do Trabalho</b>	<b>4</b>
<b>2</b>	<b>DESENVOLVIMENTO</b>	<b>6</b>
<b>2.1</b>	<b>Fundamentação Teórica</b>	<b>7</b>
<b>2.1.1</b>	Modelo Matemático Braço do Aeropêndulo	8
<b>2.1.2</b>	Modelo Matemático do Motor CC Série	10
<b>2.1.2.1</b>	Linearização do modelo do Motor CC Série	14
<b>2.1.3</b>	Junção dos subsistemas	16
<b>2.2</b>	<b>Implementação do Protótipo</b>	<b>17</b>
<b>2.2.1</b>	Protótipo Aeropêndulo	17
<b>2.2.2</b>	Parte estrutural do sistema	17
<b>2.2.3</b>	Parte Elétrica do sistema	19
<b>2.2.3.1</b>	Potenciômetro $50k\Omega$	19
<b>2.2.3.2</b>	Placa de desenvolvimento Esp32	19
<b>2.2.3.3</b>	Fonte Chaveada 2A, 5V, 25W	20
<b>2.2.3.4</b>	Módulo Driver L298n	21
<b>2.2.3.5</b>	Conjunto (suporte motor cw/ccw hélice) para drones fpv racing quadcopter	21
<b>2.2.3.6</b>	Componentes eletrônicos (resistivos e capacitivos)	22
<b>2.2.4</b>	Montagem do Protótipo	23
<b>2.2.4.1</b>	Parte Física	23

2.2.4.2	Parte Elétrica . . . . .	23
<b>2.3</b>	<b>Desenvolvimento dos Softwares . . . . .</b>	<b>24</b>
2.3.1	Linguagens Python, C e C++ . . . . .	24
2.3.2	Gêmeo Digital . . . . .	25
2.3.2.1	Biblioteca VPython . . . . .	26
2.3.2.2	Arquitetura do Gêmeo Digital . . . . .	26
2.3.3	Interface Gráfica para Configuração, visualização e aquisição de dados da Planta	29
2.3.3.1	Biblioteca CustomTkinter e Matplotlib . . . . .	30
2.3.3.2	Biblioteca PySerial, NumPy e Pandas . . . . .	31
2.3.3.3	Descrição das partes da interface gráfica . . . . .	32
2.3.4	Firmware do microcontrolador . . . . .	34
2.3.4.1	Biblioteca ler_escrever_serial . . . . .	34
2.3.4.2	Biblioteca referencia . . . . .	34
2.3.4.3	Biblioteca conversor . . . . .	35
2.3.4.4	Biblioteca controlador_pid . . . . .	35
2.3.4.5	Arquivo Principal mian.cpp . . . . .	35
<b>2.4</b>	<b>Fluxograma do Laboratório Virtual . . . . .</b>	<b>36</b>
<b>3</b>	<b>RESULTADOS E DISCUSSÕES . . . . .</b>	<b>37</b>
<b>3.1</b>	<b>Desenvolvimento do Protótipo e Softwares . . . . .</b>	<b>37</b>
<b>3.2</b>	<b>Identificação de sistema aplicado ao Aeropêndulo . . . . .</b>	<b>37</b>
3.2.0.1	Aplicação do Sinal PRBS na Entrada do Sistema . . . . .	37
3.2.0.2	Sinal de Saída Gerado a Partir do Sinal PRBS de Entrada . . . . .	38
3.2.1	Identificação Usando Mínimos Quadrados . . . . .	38
3.2.1.1	Passos para realização da identificação . . . . .	39
3.2.1.2	Identificação Usando Função de Transferência Discreta de Segunda, Quinta e Décima Ordem . . . . .	39
<b>3.3</b>	<b>Ensaio em Malha Fechada com Controlador PID . . . . .</b>	<b>45</b>
3.3.1	Onda Quadrada . . . . .	45
3.3.2	Onda Dente de Serra . . . . .	47
<b>4</b>	<b>CONCLUSÃO . . . . .</b>	<b>48</b>
<b>4.1</b>	<b>Considerações Finais . . . . .</b>	<b>48</b>
<b>4.2</b>	<b>Trabalhos Futuros . . . . .</b>	<b>49</b>
<b>APÊNDICES . . . . .</b>		<b>49</b>
<b>A</b>	<b>REPOSITÓRIO NO GITHUB . . . . .</b>	<b>50</b>
<b>REFERÊNCIAS . . . . .</b>		<b>51</b>



# INTRODUÇÃO

---

---

## 1.1 Justificativa

Os sistemas de controle desempenham um papel fundamental na sociedade contemporânea, permeando uma ampla gama de aplicações ao nosso redor. Desde o lançamento de foguetes e o decolamento do ônibus espacial até processos de usinagem automatizados, onde peças metálicas são trabalhadas envoltas em jatos de água de resfriamento, até veículos autônomos que distribuem materiais em oficinas aeroespaciais, a presença e a influência dos sistemas de controle automático são evidentes, (NISE, 2013). É crucial ressaltar que, ao realizar análises e projetar controladores eficazes para tais sistemas, é necessário realizar uma abstração desses sistemas físicos por meio de equações matemáticas. Essa prática, embora essencial, pode tornar a interpretação desafiadora para iniciantes na área devido à complexidade intrínseca das formulações matemáticas envolvidas.

Além disso, a implementação de controladores requer competência em diferentes áreas da engenharia, tais como eletrônica analógica e digital, programação, processamento de sinais, circuitos elétricos, entre outras. Dessa forma, torna-se necessário integrar conhecimentos multidisciplinares para a implementação bem-sucedida de controladores em sistemas reais.

Um sistema de controle é composto por subsistemas e processos, concebidos com a finalidade de alcançar uma saída desejada com um desempenho específico, considerando uma entrada predefinida, (NISE, 2013).

A abordagem convencional no ensino das disciplinas teóricas em cursos de Engenharia representa um desafio significativo, dada a dificuldade dos alunos em estabelecer conexões entre o conteúdo teórico e a aplicação prática nos sistemas físicos. A introdução de tecnologias de apoio tem proporcionado avanços, mas persistem desafios, como a carência de estrutura operacional e a necessidade de atualização nas metodologias de ensino. A incorporação de simulações

computacionais aliadas a animações emerge como uma estratégia promissora para acelerar esse processo, oferecendo uma ponte eficaz entre a teoria e a compreensão prática, ([COTA, 2023](#)).

No entanto, no estudo de sistemas de controle, é comum que os discentes enfrentem desafios em seu primeiro contato com a área, especialmente devido à necessidade de aplicar abstrações matemáticas para representar a dinâmica de sistemas físicos. Essa etapa inicial pode parecer complexa, porém é crucial para a compreensão e domínio dos conceitos fundamentais envolvidos na análise e controle de sistemas.

Uma das principais razões pelas quais os estudantes encontram dificuldades é a transição do mundo físico para o mundo matemático abstrato, onde os sistemas são modelados por equações diferenciais, transformadas de Laplace e outros formalismos matemáticos. Para muitos, essa mudança pode parecer distante da realidade observada, o que pode causar alguma resistência inicial.

Assim, buscando superar essa barreira de aprendizagem, é fundamental desenvolver métodos que possibilitem aos discentes visualizar a dinâmica desses sistemas na prática. Uma abordagem promissora é a utilização de protótipos, que permitem aos alunos observar a dinâmica analisada matematicamente no mundo real. Essa aplicação prática fornece uma conexão mais tangível entre os conceitos abstratos e suas aplicações concretas, tornando o aprendizado mais envolvente e compreensível.

Adicionalmente, o uso de simuladores pode ser altamente benéfico. Ao inserir as respostas obtidas por meio dos modelos matemáticos como entrada nos simuladores, o processo de aprendizado integra ferramentas matemáticas e de visualização. Dessa forma, os alunos podem interagir com os sistemas em diferentes cenários, observando como as variáveis influenciam o comportamento dos sistemas de controle. Essa abordagem interativa e experimental ajuda a solidificar conceitos e aprimorar a compreensão do funcionamento desses sistemas complexos.

Ao combinar a teoria matemática com a prática através de protótipos e simuladores, o processo de aprendizado torna-se mais fluido e estimulante. Os alunos podem perceber a relevância das abstrações matemáticas na resolução de problemas reais, o que reduzirá a resistência inicial e aumentará o interesse pela área de sistemas de controle.

A proposta deste trabalho consiste no desenvolvimento de um protótipo que utiliza um conjunto de softwares para criar um laboratório virtual. Essa abordagem possibilita que os estudantes construam seu próprio laboratório de maneira ágil e flexível, utilizando componentes acessíveis e softwares de código aberto, sem custos adicionais. Dessa forma, os estudantes têm a oportunidade de aprimorar seus conhecimentos e aprofundar tanto a teoria quanto a prática de forma complementar, resultando em uma base de aprendizado mais sólida.

O projeto comprehende um protótipo de Aeropêndulo composto por firmware para o microcontrolador, software de usuário e um simulador 3D. Este protótipo foi concebido com a premissa de ter o menor custo possível, visando proporcionar aos interessados uma implementação

acessível com gastos mínimos em componentes. O algoritmo do firmware foi desenvolvido em linguagens C e C++, utilizando o Framework do Arduino em conjunto com o PlatformIO e o microcontrolador ESP32.

A interface do usuário foi projetada com o objetivo de simplificar a interação em tempo real, permitindo ajustes nos parâmetros do sinal de referência durante a execução do sistema. Por fim, foi criado um gêmeo digital do protótipo físico, possibilitando a observação da dinâmica do sistema físico por meio de um simulador 3D. Este simulador replica o comportamento do protótipo em tempo real, proporcionando uma integração eficaz entre o mundo real e o computacional.

## 1.2 Objetivos

### 1.2.1 Objetivos Gerais

Este trabalho tem como objetivo desenvolver um laboratório virtual abrangente para o estudo de sistemas de controle. Para alcançar esse propósito, será criado um projeto que integra um protótipo, simulador e uma interface gráfica, possibilitando sua utilização na investigação de diversos aspectos e subáreas relacionadas a sistemas de controle. Isso inclui a identificação de sistemas, o desenvolvimento de controladores, a otimização de controladores, bem como a aplicação de Inteligência Artificial em sistemas de controle, entre outros tópicos.

Além disso, a proposta envolve a aplicação prática dos conhecimentos adquiridos durante a graduação, unindo técnicas de sistemas de controle, eletrônica analógica e digital, programação, física e cálculo em uma configuração de sistema físico. O principal objetivo é criar um ambiente que permita ao pesquisador aplicar e aprimora seus conhecimentos e observar o comportamento da dinâmica do sistema em um ambiente real. Para realizar essa tarefa, será necessário combinar tecnologias de diversas disciplinas do curso de Engenharia Elétrica, tornando o projeto ainda mais intrigante e desafiador.

### 1.2.2 Objetivos Específicos

- **Desenvolver o protótipo do Aeropêndulo:** Projetar a estrutura física e elétrica do sistema;
- **Desenvolver um simulador 3D (Gêmeo Digital):** programar o simulador usando a linguagem de programação Python em conjunto com a biblioteca VPython;
- **Interface de Usuário:** Desenvolver uma interface gráfica com menu para manipular o comportamento do sistema em tempo real, salvar dados de ensaio e testar o sistema em malha fechada;
- **Aplicar Identificação de sistema a Planta:** Obter um modelo da planta usando função de transferência discreta por identificação de sistemas usando o método dos mínimos quadrados;

- **Testar o sistema em Malha Fechada:** Implementar um controlador PID junto do firmware e realizar testes para validar o sistema em malha fechada;

## 1.3 Escopo do Trabalho

O projeto parte de uma modelagem matemática usando como base os princípios da física newtoniana com o intuito de demostrar que para sistemas relativamente complexos, essa técnica de modelagem pode se tornar trabalhosa e por muitas vezes impraticáveis, a partir dessa premissa parte-se para o desenvolvimento do protótipo, o objetivo está na utilização do sistema físico para aplicar o método de identificação de sistema que consiste em obter um modelo matemático, que descreva a dinâmica do sistema físico de forma aproximada, a partir dos dados de entrada e saída do protótipo.

No processo de condução dos ensaios e na aquisição dos dados essenciais para a realização dos estudos, o projeto inclui o desenvolvimento de uma interface gráfica. Esta interface capacita o pesquisador a executar ensaios e a armazenar os dados obtidos, viabilizando a subsequente modelagem e análise do comportamento do sistema em estudo. Adicionalmente, a interface é equipada com um menu que permite a configuração em tempo real dos ensaios para diferentes sinais de entrada, além da configuração dos parâmetros de frequência, amplitude e offset. Essa funcionalidade dinâmica e flexível aprimora significativamente a condução da pesquisa, tornando-a mais produtiva e eficiente.

Além da interface gráfica executada no computador, é fundamental o desenvolvimento de um firmware que gerencie aspectos cruciais, como a comunicação, aquisição de dados e controle do Aeropêndulo. Essas funcionalidades são implementadas por um microcontrolador. Este microcontrolador não apenas estabelece a comunicação com a interface gráfica através da conexão USB, mas também desempenha um papel vital na geração dos sinais de entrada para a planta e na leitura do ângulo do braço do Aeropêndulo.

Além disso, o projeto incorpora um gêmeo digital do sistema físico, na forma de um simulador 3D do Aeropêndulo. Essa abordagem permite a integração da dinâmica do sistema real com o ambiente computacional, possibilitando a recriação em tempo real da planta de forma virtual. Essa simulação em 3D não apenas enriquece a experiência de aprendizado, mas também oferece aos estudantes a oportunidade de se envolver em um processo de aprendizado prático que se assemelha significativamente às condições do mundo real. Isso resulta em um processo educacional mais imersivo e eficaz, capacitando os alunos a compreender e explorar a complexidade dos sistemas de controle de maneira mais aprofundada.

A seção 2.1 desenvolve a fundamentação teórica focando na modelagem matemática do sistema, a seção 2.2 implementa o protótipo de Aeropêndulo mostrando passo a passo o processo de desenvolvimento até a concepção da planta, para o desenvolvimento dos softwares a seção

2.3 descreve as etapas do desenvolvimento das aplicações, finalizando o capítulo 2 a seção 2.4 descreve as partes do laboratório a partir de um fluxograma do sistema.

O capítulo 3 aborda os resultados e discussões obtidos ao término do projeto. Na seção 3.2, emprega-se o método de identificação de sistemas para validar a estrutura de obtenção e armazenamento dos dados de ensaio. Adicionalmente, a seção 3.3 implementa um controlador PID simples com o propósito de validar a infraestrutura física e de software desenvolvida para os testes de controladores na planta física.

Por fim, o capítulo 4 consolida os avanços alcançados no projeto, destacando perspectivas futuras para sua aplicação. Na seção 4.1, são apresentadas considerações finais sobre a implementação realizada. Além disso, a seção 4.2 oferece uma visão prospectiva sobre possíveis trabalhos futuros que podem adotar o projeto desenvolvido neste trabalho como ambiente de estudo.

CAPÍTULO  
**2**

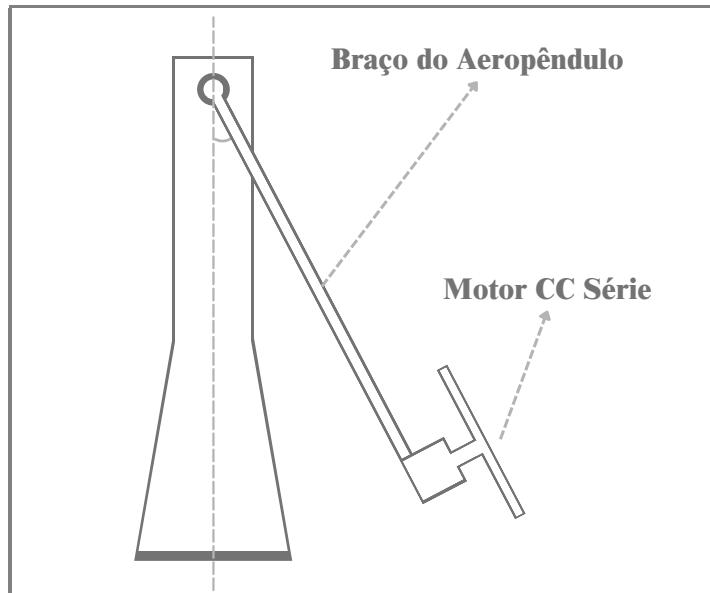
---

# DESENVOLVIMENTO

---

Os sistemas mecânicos têm como uma de suas características os graus de liberdade, o Aeropêndulo tem um grau de liberdade, sendo o ângulo  $\theta$ , assim o sistema tem apenas uma variável a ser controlada a partir do empuxo gerado pelas hélices, que por sua vez depende da velocidade angular do eixo do motor CC Série acoplado ao braço do Aeropêndulo. dessa forma, podemos analisar o sistema a partir da entrada e da saída, sendo a entrada a tensão no motor e a saída o ângulo  $\theta$  do braço do Aeropêndulo.

Figura 1 – Diagrama esquemático do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

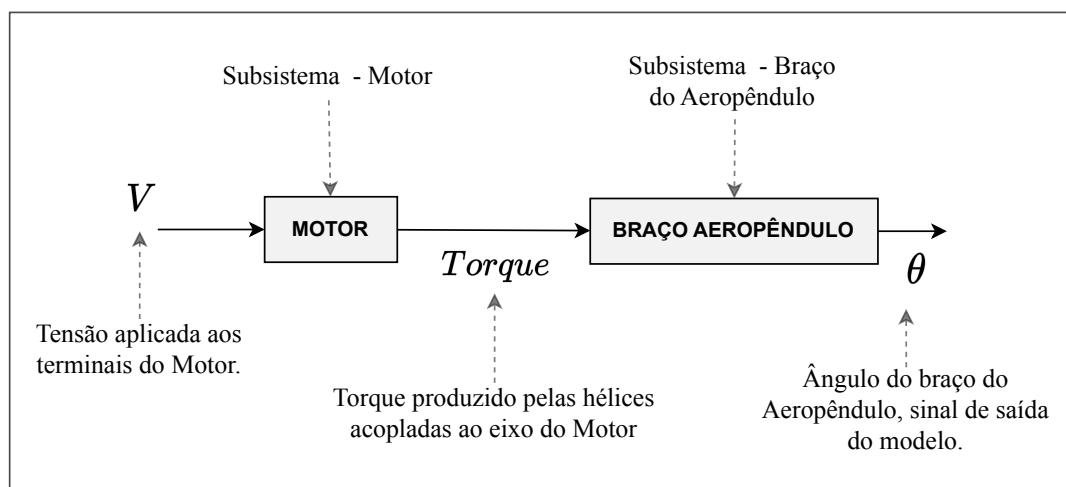
## 2.1 Fundamentação Teórica

O processo que envolve modelagem de sistemas físicos em termos de equações matemáticas é uma das partes mais importante no estudo de sistemas de controle. Segundo OGATA (2014, p. 11), O modelo matemático de um sistema dinâmico é definido como um conjunto de equações que representa a dinâmica do sistema com precisão ou, pelo menos, razoavelmente bem.

A construção de modelos matemáticos adequados é fundamental na análise de sistemas de controle, pois a dinâmica de diversos sistemas, como mecânicos, elétricos, térmicos, econômicos, biológicos, entre outros, pode ser expressa por meio de equações diferenciais. Tais equações são derivadas das leis físicas que governam cada sistema específico, como as leis de Newton para sistemas mecânicos e as leis de Kirchhoff para sistemas elétricos. Esta abordagem ressalta a importância crucial de compreender as bases matemáticas subjacentes para uma análise abrangente dos sistemas de controle, conforme destacado por OGATA (2014, p. 11).

Para realizar a modelagem do Aeropêndulo pode-se aplicar diferentes métodos, uma abordagem inicial para modelar sistemas físicos consiste em empregar as leis fundamentais da física. Além disso, em situações mais complexas, é viável decompor o sistema em subsistemas menores e, em seguida, desenvolver modelos para cada um deles. Por fim, é possível conectar esses modelos, de forma a obter uma representação aproximada do sistema real. Dessa maneira, podemos obter uma compreensão mais aprofundada e abrangente da dinâmica do sistema em questão. A Figura 2 mostra um diagrama da representação do Aeropêndulo como um conjunto de subsistemas.

Figura 2 – Subsistemas do Aeropêndulo.

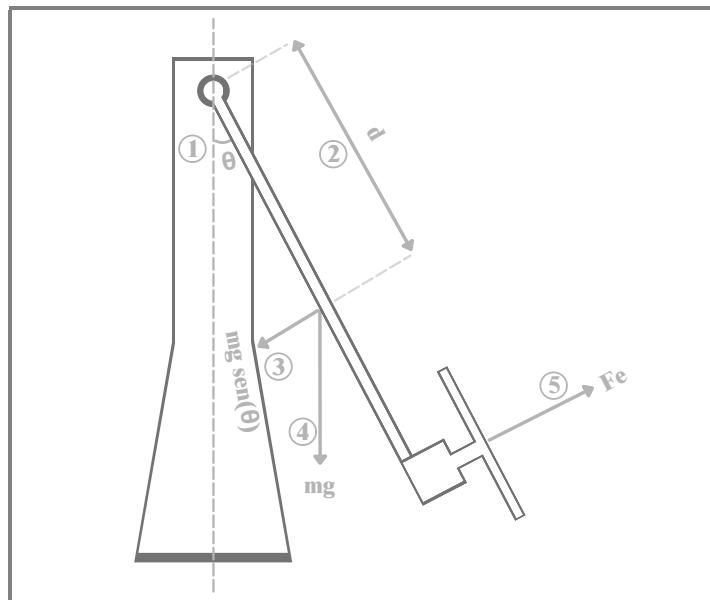


Fonte: elaborado pelo autor (2023).

### 2.1.1 Modelo Matemático Braço do Aeropêndulo

Como previamente abordado, é possível segmentar o sistema em dois subsistemas distintos: o motor CC Série e o braço do Aeropêndulo. Nesta seção, o foco estará concentrado na modelagem do braço do Aeropêndulo, utilizando como referência o trabalho de ([MOHAMMADBAGHERI; YAGHOBI, 2011](#)) para a obtenção da equação que descreve a dinâmica desse componente.

Figura 3 – Diagrama esquemático do Braço do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

O modelo matemático do braço do Aeropêndulo é derivado a partir das leis de Newton e do momento angular, como mostra ([MOHAMMADBAGHERI; YAGHOBI, 2011](#)). assim, se obtém a equação 2.1.

$$F_e = J_b \ddot{\theta} + c \dot{\theta} + mgd \sin \theta \quad (2.1)$$

Onde:

- $F_e$ : Empuxo gerado pela hélice
- $J_b$ : Momento de inércia do Braço
- $\theta$ : posição angular do Aeropêndulo
- $c$ : coeficiente de amortecimento viscoso
- $m$ : massa do Aeropêndulo
- $d$ : a distância entre o centro de massa e o ponto de pivô

A entrada do subsistema do braço do Aeropêndulo é a força de empuxo proporcionada pela hélice, porém, o modelo do motor CC Série tem como saída a velocidade angular, dessa forma é preciso encontrar uma relação entre a velocidade  $\dot{\omega}$  e o empuxo  $F_e$ , essa relação é não linear como mostra [xx],  $F_e = K_m \dot{\omega}^2$ , porém é possível aproximar por uma relação linear,  $F_e = K_m \dot{\omega}$ . Com isso, pode-se relacionar a velocidade angular  $\dot{\omega}$  com o empuxo  $F_e$  gerado pela hélice do Aeropêndulo.

O modelo encontrado tem uma parcela não linear dada por  $\sin \theta$ , para aplicar técnicas de projeto de controle é preciso obter o modelo linearizado da planta, isso pode ser feito considerando  $\sin \theta \approx \theta$  para pequenas variações em torno de  $\theta$ . dessa forma, temos a seguinte linearização:

$$F_e = J_b \ddot{\theta} + c \dot{\theta} + mgd\theta \quad (2.2)$$

Aplicando a transformada de Laplace para encontrar a função de transferência do subsistema, tem-se:

$$F_e(s) = J_b \theta(s)s^2 + c\theta(s)s + mgd\theta(s) \quad (2.3)$$

$$F_e(s) = (J_b s^2 + cs + mgd)\theta(s) \quad (2.4)$$

$$\frac{\theta(s)}{F_e(s)} = \frac{1}{J_b s^2 + cs + mgd} \quad (2.5)$$

Agora que foi obtido o modelo do braço do Aeropêndulo, pode-se usar a relação linearizada  $F_e = K_m \dot{\omega}$  para usar a saída do modelo do motor cc série como entrada do modelo do braço do Aeropêndulo, como mostrado na equação 2.8.

$$\frac{\theta(s)}{K_m \dot{\omega}(s)} = \frac{1}{J_b s^2 + cs + mgd} \quad (2.6)$$

$$\frac{\theta(s)}{\dot{\omega}(s)} = \frac{K_m}{J_b s^2 + cs + mgd} \quad (2.7)$$

$$H(s) = \frac{\theta(s)}{\dot{\omega}(s)} = \frac{\frac{K_m}{J_b}}{s^2 + \frac{c}{J_b}s + \frac{mgd}{J_b}} \quad (2.8)$$

A equação 2.8 representa a função de transferência que descreve o comportamento dinâmico do braço do Aeropêndulo. Esta expressão matemática encapsula as relações fundamentais entre as variáveis de entrada/saída do sistema, oferecendo uma visão abrangente e quantitativa da resposta do braço às diferentes condições e estímulos. Ao analisar a função de transferência, podemos compreender como as mudanças nas entradas afetam as saídas.

### 2.1.2 Modelo Matemático do Motor CC Série

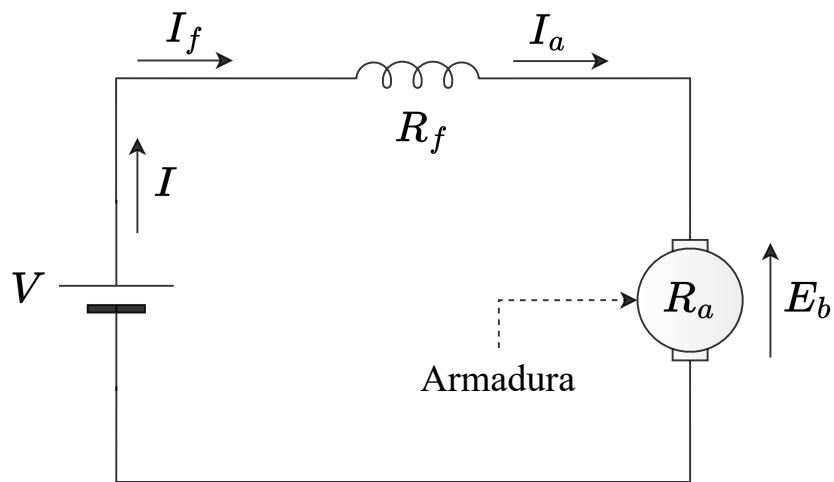
Os motores CC série tem como principal característica possuir o enrolamento de campo em série com o enrolamento de armadura, essa configuração resulta em um motor com torque de partida alto, porém, o torque reduz a medida que a velocidade aumenta devido ao aumento da Força Eletromotriz FEM. Por conta desse aumento de FEM os motores CC Séries tem uma regulação de velocidade ruim, quando se aumenta a carga no eixo do motor a velocidade é reduzida que por sua vez reduz a FEM e então o torque aumenta para conseguir atuar na carga.

No contexto dos motores série, observamos que o aumento da carga resulta em incrementos na corrente, na Força Magnetomotriz (FMM) da armadura e no fluxo do campo do estator, contanto que o ferro não atinja a saturação completa. À medida que o fluxo aumenta com a carga, a velocidade do motor série deve diminuir para manter um equilíbrio adequado entre a tensão aplicada e a força contraeletromotriz. Além disso, o aumento na corrente de armadura, ocasionado pelo acréscimo no conjugado, é menor em comparação com o motor em derivação, devido ao aumento no fluxo. Dessa forma, o motor série é caracterizado como um motor de velocidade variável., [UMANS \(2014, p. 410\)](#).

No entanto, mesmo motores cc série com dimensões reduzidas geram torques altos com baixo consumo de corrente. Visando melhorar seu desempenho, é possível projetar controladores de malha fechada capazes de tornar esses motores mais eficientes na regulação de velocidade.

Para realizar a modelagem o motor CC série, foi usado a base teórica de ([LICEAGA-CASTRO et al., 2017](#)). A Figura 4 mostra um diagrama da configuração do motor CC Série, no qual o enrolamento de campo está conectado em série com o enrolamento de armadura, dessa forma, a corrente de campo é igual a corrente de armadura  $i = i_f = i_a$ .

Figura 4 – Motor CC Série.

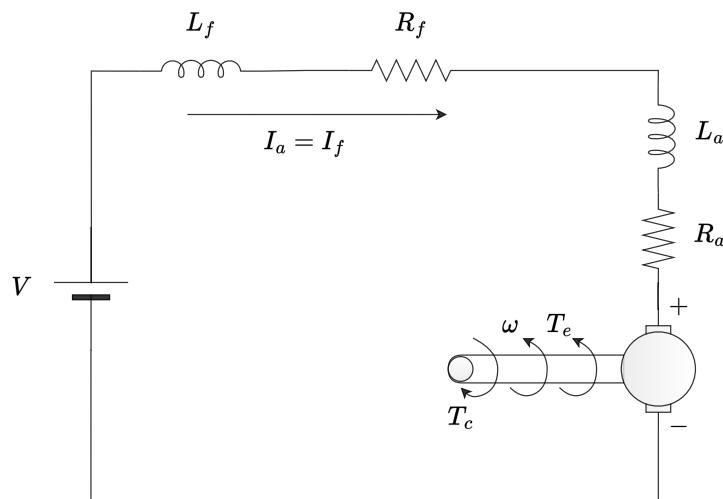


Fonte: elaborado pelo autor (2023).

Na Figura 5, mostra o diagrama eletromecânico do motor, nele pode-se observar que os

componentes elétricos estão todos em série, no qual o enrolamento de campo possui uma parte resistiva e outra indutiva, assim como o enrolamento de armadura, já a parte mecânica possui uma velocidade angular dada por  $\dot{\omega}$ , torque eletromagnético do motor dado por  $T_e$  e torque da Carga  $T_c$ .

Figura 5 – Diagrama Elétrico/Mecânico Motor CC Série.



Fonte: elaborado pelo autor (2023).

Primeiramente a parte mecânica do motor será modelada, um motor cc série é composto por uma parte rotativa "armadura", de modo que, essa parte, gera um momento de inércia  $J_m$  no eixo do motor e um fator de amortecimento viscoso  $b$ , além disso, o eixo possui uma velocidade angular  $\dot{\omega}$ .

Assim, a equação 2.9, obtida a partir de ([LICEAGA-CASTRO et al., 2017](#)), descreve o modelo mecânico do motor CC Série.

$$J_m \ddot{\omega}(t) = T_e(t) - b\dot{\omega}(t) - T_c(t) \quad (2.9)$$

Ao expressar o torque  $T_e(t)$  em função das outras variáveis, obtém-se a equação 2.10.

$$T_e(t) = J_m \ddot{\omega}(t) + b\dot{\omega}(t) + T_c(t) \quad (2.10)$$

Onde:

- $T_e$ : Torque Eletromagnético produzido pelo Motor;
- $J_m$ : Momento de Inércia do Eixo do Motor;
- $\dot{\omega}$ : Aceleração Angular do Eixo do Motor;

- $\dot{\omega}$ : Velocidade Angular do Eixo do Motor;
- $b$ : Fator de Amortecimento Viscoso;
- $T_c$ : Torque de Carga.

Tanto a Força Eletromotriz  $E_A(t)$  quanto o Torque Eletromagnético  $T_e(t)$  dependem do fluxo magnético do entreferro  $\Phi$ , com isso, tem-se as equações 2.11 e 2.12.

$$E_a(t) = \dot{\omega}(t)\Phi(i) \quad (2.11)$$

$$T_e(t) = i(t)\Phi(i) \quad (2.12)$$

O Fluxo magnético depende da corrente  $i(t)$ , assim, as equações 2.9 e 2.10 são não-lineares. Além disso, pode-se aproximar o fluxo  $\Phi$  por uma relação linear,  $K_0$ , ao desprezar a saturação magnética.

$$\Phi(i) = K_0i(t) \quad (2.13)$$

A constante  $K_0$  é a indutância mútua entre a armadura e o enrolamento de campo. Agora pode-se encontrar o modelo não-linear da parte mecânica do Motor CC Série, substituindo 2.13 em 2.12, obtém-se a expressão 2.15:

$$T_e(t) = i(t)K_0i(t) \quad (2.14)$$

$$T_e(t) = K_0i^2(t) \quad (2.15)$$

Substituindo 2.15 em 2.10, é possível encontrar o modelo da parte mecânica do motor CC série, assim, tem-se a expressão 2.16.

$$K_0i^2(t) = J_m\ddot{\omega}(t) + b\dot{\omega}(t) + T_c(t) \quad (2.16)$$

Para a parte elétrica, utilizou-se a lei de Kirchhoff das tensões para modelar o sistema. assim como a parte mecânica, o modelo da parte elétrica foi baseado de ([LICEAGA-CASTRO et al., 2017](#)), o motor em questão é um motor de corrente contínua CC série, Dessa forma,  $i_a = i_f$ , portanto, obtém-se a expressão 2.17.

$$V(t) = (R_a + R_f)i(t) + (L_a + L_f)\frac{di}{dt} + E_a \quad (2.17)$$

Onde:

- $V$ : Tensão da Fonte;
- $R_a$ : Resistência da Armadura;
- $R_f$ : Resistência de Campo;
- $i_a$ : Corrente da Armadura;
- $i_f$ : Corrente de Campo;
- $E_A$ : Tensão Contro Eletromotriz Gerada pela Armadura;
- $L_a$ : Impedância da Armadura;
- $L_f$ : Impedância de Campo.

Como os componentes elétricos então em série, pode-se obter uma resistência total assim como uma indutância:

$$R = R_a + R_f \quad (2.18)$$

$$L = L_a + L_f \quad (2.19)$$

$$V(t) = Ri(t) + L \frac{d}{dt} i(t) + E_a \quad (2.20)$$

Substituindo 2.13 em 2.11, obtém-se a equação 2.21.

$$E_a(t) = \dot{\omega}(t)K_0i(t) \quad (2.21)$$

Por fim, pode-se encontrar a equação que descreve a parte elétrica do sistema ao substituir 2.21 em 2.20, assim, obtendo a equação 2.22.

$$V(t) = Ri(t) + L \frac{d}{dt} i(t) + \dot{\omega}(t)K_0i(t) \quad (2.22)$$

Dessa forma, as equações que modelam um Motor CC série são expressas por 2.23 e 2.24, em que 2.23 esta relacionada a parte mecânica e 2.24 a parte elétrica.

$$K_0i^2(t) = J_m\ddot{\omega}(t) + b\dot{\omega}(t) + T_c(t) \quad (2.23)$$

$$V(t) = Ri(t) + L \frac{d}{dt} i(t) + \dot{\omega}(t)K_0i(t) \quad (2.24)$$

### 2.1.2.1 Linearização do modelo do Motor CC Série

Para projetos de controle lineares é preciso linearizar o modelo encontrado, para isso, vamos reorganizar as equações 2.23 e 2.24, assim obtêm-se as equações 2.25 e 2.26:

$$\ddot{\omega}(t) = \frac{K_0}{J_m} i^2(t) - \frac{b}{J_m} \dot{\omega}(t) - \frac{1}{J_m} T_c(t) \quad (2.25)$$

$$\frac{d}{dt} i(t) = \frac{R}{L} i(t) - \frac{K_0}{L} \dot{\omega}(t) i(t) + \frac{1}{L} V(t) \quad (2.26)$$

reescrevendo as equações de estados na forma matricial,

$$x_1 = \dot{\omega} \quad (2.27)$$

$$x_2 = i \quad (2.28)$$

Coeficientes da equação de estado 2.25.

$$a_1 = \frac{K_0}{J_m}; \quad a_2 = \frac{b}{J_m}; \quad a_3 = \frac{1}{J_m} \quad (2.29)$$

Coeficientes da equação de estado 2.26.

$$b_1 = \frac{R}{L}; \quad b_2 = \frac{K_0}{L}; \quad b_3 = \frac{1}{L} \quad (2.30)$$

Representação no espaço de estados do motor CC Série não-linear

$$\dot{x}_1 = a_1 x_2^2 - a_2 x_1 - a_3 T_c \quad (2.31)$$

$$\dot{x}_2 = -b_1 x_2 - b_2 x_1 x_2 + b_3 V \quad (2.32)$$

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} a_1 x_2^2 - a_2 x_1 - a_3 T_c \\ -b_1 x_2 - b_2 x_1 x_2 + b_3 V \end{bmatrix} = f(x, u) \quad (2.33)$$

O ponto de equilíbrio  $(x_1^0, x_2^0)$  das equações 2.31 e 2.32 é encontrado zerando as derivadas, como mostrado em 2.34 e 2.35.

$$x_2^0 = \sqrt{\frac{a_2 x_1^0 + a_3 T_c}{a_1}} \quad (2.34)$$

$$V = \frac{x_2^0(b_1 + b_2 x_1^0)}{b_3} \quad (2.35)$$

A linearização de 2.31 e 2.32 em torno do ponto de equilíbrio é obtida encontrando o jacobiano das mesmas, assim se obtêm as matrizes  $A$  e  $B$ .

$$\dot{x} = Ax + Bu \quad (2.36)$$

$$y = Cx \quad (2.37)$$

Em que,

$$A = \left. \frac{\partial f(x, y)}{\partial x} \right|_{x_1^0, x_2^0} = \begin{bmatrix} -a_2 & 2a_1 x_2^0 \\ -b_2 x_2^0 & -(b_1 + b_2 x_1^0) \end{bmatrix} \quad (2.38)$$

$$B = \left. \frac{\partial f(x, y)}{\partial u} \right|_{x_1^0, x_2^0} = \begin{bmatrix} -a_3 & 0 \\ 0 & b_3 \end{bmatrix} \quad (2.39)$$

$$C = [1, 0] \quad (2.40)$$

$$(2.41)$$

Onde,

$$u = \begin{bmatrix} T_c \\ V \end{bmatrix} \quad (2.42)$$

$$y = \dot{\omega}(t) \quad (2.43)$$

Se o torque de carga for considerado zero,  $T_c = 0$ , temos as seguintes expressões,

$$x_2^0 = \sqrt{\frac{a_2 x_1^0}{a_1}}; \quad V = \frac{x_2^0(b_1 + b_2 x_1^0)}{b_3}; \quad B = \begin{bmatrix} 0 \\ b_3 \end{bmatrix} \quad (2.44)$$

Agora é possível encontrar a função de transferência  $G(s)$  associada as matrizes de estados  $A, B, C$  linearizadas, para isso usa-se a expressão 2.45.

$$G(s) = \frac{\dot{\omega}(s)}{V(s)} = C(sI - A)^{-1}B \quad (2.45)$$

Substituindo as matrizes  $A$ ,  $B$  e  $C$  em 2.45, chega-se a expressão 2.46.

$$G(s) = \frac{\dot{\omega}(s)}{V(s)} = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \left( s \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} -a_2 & 2a_1x_2^0 \\ -b_2x_2^0 & -(b_1 + b_2x_1^0) \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} -a_3 & 0 \\ 0 & b_3 \end{bmatrix} \quad (2.46)$$

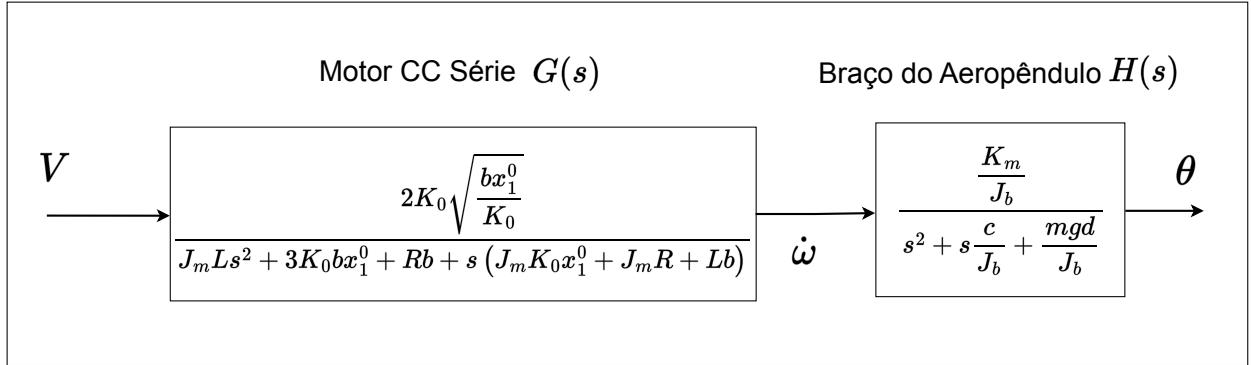
Assim, obtém-se a função de transferência linearizada em função dos parâmetros do motor CC Série, como mostra a equação 2.47.

$$G(s) = \frac{\dot{\omega}(s)}{V(s)} = \frac{2K_0\sqrt{\frac{bx_1^0}{K_0}}}{J_m L s^2 + (J_m K_0 x_1^0 + J_m R + Lb) s + 3K_0 b x_1^0 + Rb} \quad (2.47)$$

### 2.1.3 Junção dos subsistemas

A partir dos modelos encontrados nas subseções 2.1.2 e 2.1.1 pode-se obter o modelo completo do sistema unindo os subsistemas como mostra a figura 6. com isso, é possível modelar o Aeropêndulo tendo como entrada a tensão no motor CC série e como saída o ângulo do braço.

Figura 6 – Diagrama da junção dos subsistemas do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

A modelagem usando as leis físicas permite uma interpretação dos parâmetros da planta e ajuda a entender a influência deles na dinâmica do sistema, no entanto, ao utilizar esse método de modelagem surgi uma complexidade em obter os valores numéricos de alguns desses parâmetros, por exemplo, o Momento de Inércia do Eixo do Motor  $J_m$  se torna complicado de se obter, assim como o Coeficiente de Amortecimento Viscoso  $c$  do pivô do braço do Aeropêndulo entre outros parâmetros, com isso, surgi um método alternativo que permite modelar um sistema usando dados de ensaios chamado de Identificação de Sistema, esse método usando os dados de entrada e saída e aproxima um modelo, com isso se obtém uma função de transferência discreta.

## 2.2 Implementação do Protótipo

### 2.2.1 Protótipo Aeropêndulo

O projeto para o desenvolvimento do protótipo consiste em uma estrutura mecânica e componentes eletrônicos, a Figura 7 mostra o protótipo finalizado com a parte estrutural e elétrica montada.

Figura 7 – Protótipo do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

### 2.2.2 Parte estrutural do sistema

O material escolhido para a estrutura foi o compensado, as chapas de compensado são materiais versáteis e amplamente utilizados na indústria da construção, marcenaria e artesanato. Elas são fabricadas a partir de finas lâminas de madeira, conhecidas como lâminas de folheado, coladas umas sobre as outras com fibras perpendiculares, conferindo maior estabilidade e resistência mecânica ao produto final, Figura 8.

Figura 8 – Chapas de compensado.



Fonte: elaborado pelo autor (2023).

Já para o braço do Aeropêndulo foi usando Tubo de Fibra de Vidro de Carbono 3x3x2mm, Figura 9, Este material destaca-se por sua leveza e resistência física. No caso do braço do Aeropêndulo, localizado em sua extremidade, há um conjunto motor/hélice encarregado de impulsionar a dinâmica do sistema. Para otimizar o desempenho, é essencial minimizar a massa do braço. Ao fazer isso, as forças de resistência enfrentadas pela força de empuxo serão reduzidas, resultando em uma dinâmica mais ágil e eficiente do sistema.

Figura 9 – Tubo de Fibra de Vidro de Carbono 3x3x2mm.



Fonte: elaborado pelo autor (2023).

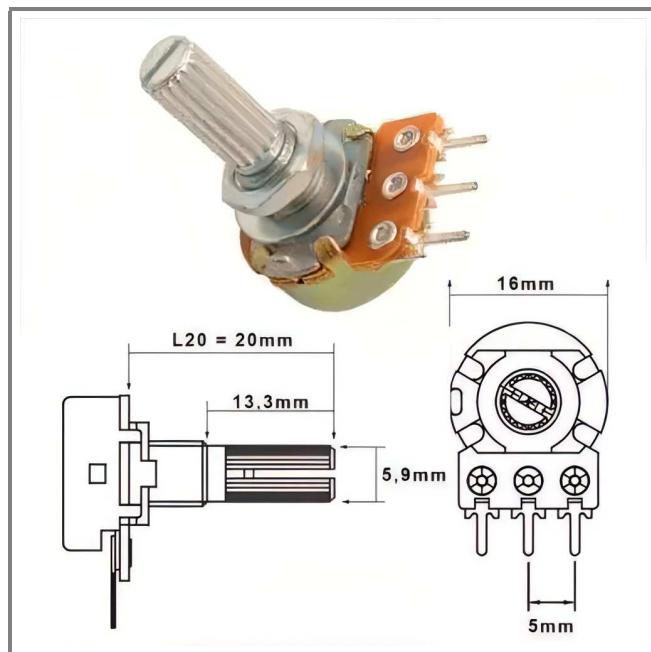
### 2.2.3 Parte Elétrica do sistema

Foram empregados os seguintes componentes eletrônicos no projeto: um Potenciômetro  $50k\Omega$ , uma Placa de desenvolvimento Esp32, um Módulo Driver L298n, um Conjunto (suporte/motor/hélice para drones fpv racing quadcopter) e componentes eletrônicos (resistor, capacitor).

#### 2.2.3.1 Potenciômetro $50k\Omega$

O potenciômetro desempenha o papel de *encoder* ao obter o ângulo de inclinação do braço do Aeropêndulo. Essa funcionalidade é viabilizada pelo fato de que, conforme o braço se movimenta, a resistência do potenciômetro se altera, resultando em uma variação na tensão registrada em seus terminais. Essa interação permite estabelecer uma correlação entre a mudança na tensão elétrica e o ângulo de inclinação do braço do Aeropêndulo de maneira precisa e controlada.

Figura 10 – Potenciômetro  $50k\Omega$ .



Fonte: elaborado pelo autor (2023).

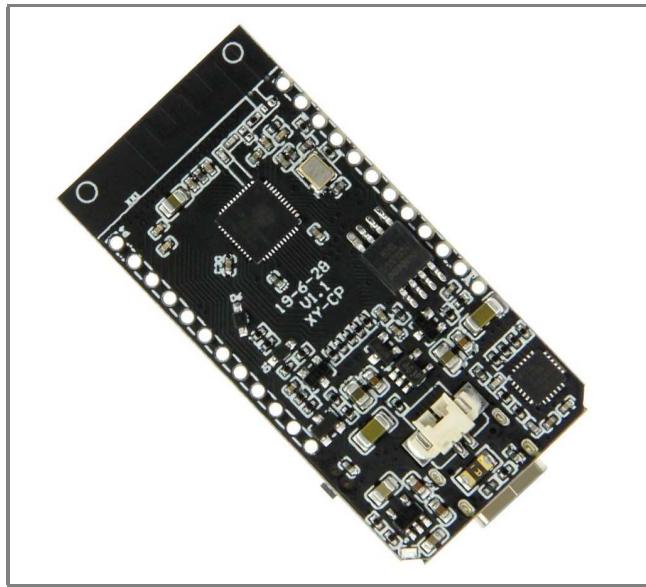
#### 2.2.3.2 Placa de desenvolvimento Esp32

A placa de desenvolvimento ESP32 é uma poderosa e versátil plataforma que integra o popular microcontrolador ESP32 desenvolvido pela empresa *Espressif*. Projetada para atender às demandas de projetos de Internet das Coisas (IoT) e aplicações de conectividade sem fio, a ESP32 oferece uma vasta gama de recursos e funcionalidades. Equipada com um processador dual-core de 32 bits, Wi-Fi, Bluetooth, interfaces GPIO, I2C, SPI e UART, bem como suporte para memória externa.

A placa desempenhará uma função central no projeto, servindo tanto para a implementação do controlador discreto quanto para a obtenção de dados em tempo real relacionados aos estados

do sistema. Esses estados incluem a leitura do sensor de ângulo do braço, a aquisição do sinal de controle e o cálculo do sinal de erro. Adicionalmente, a placa será empregada na geração dos sinais de referência necessários para o funcionamento ideal do sistema. A figura 11 ilustra a placa de desenvolvimento Esp32.

Figura 11 – Placa de desenvolvimento Esp32.



Fonte: elaborado pelo autor (2023).

#### 2.2.3.3 Fonte Chaveada 2A, 5V, 25W

Para a alimentação do motor CC série, foi usado uma fonte de alimentação de 5V/5A, Figura 12.

Figura 12 – Fonte Chaveada 2A, 5V, 25W.

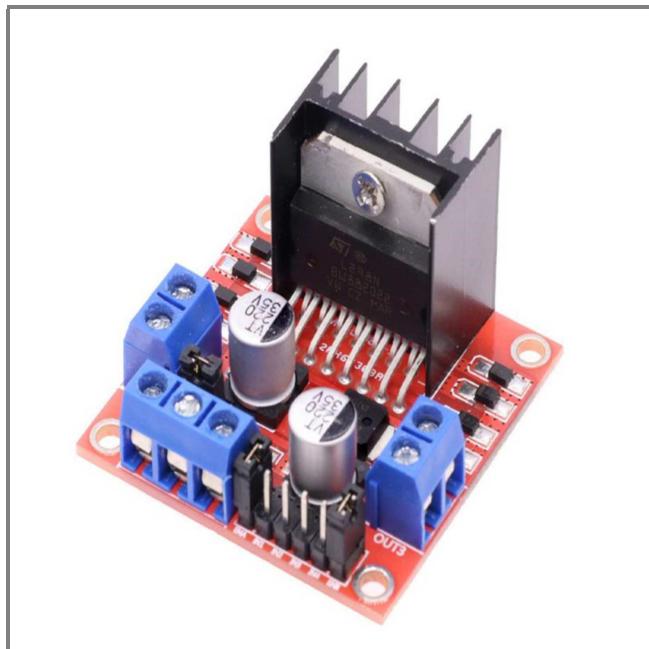


Fonte: elaborado pelo autor (2023).

### 2.2.3.4 Módulo Driver L298n

Para regular a velocidade no eixo do Motor CC série, empregou-se um Módulo Driver L298n, o qual possibilita controlar a injeção de potência entregue ao motor mediante a aplicação de um sinal PWM em sua entrada. Dessa forma, ao ajustar o sinal PWM, é possível obter uma tensão controlada aplicada de maneira precisa aos terminais do motor. A Figura 13 ilustra o componente em questão.

Figura 13 – Módulo Driver L298n.



Fonte: elaborado pelo autor (2023).

### 2.2.3.5 Conjunto (suporte motor cw/ccw hélice) para drones fpv racing quadcopter

Para obter-se a força de empuxo na extremidade do braço do Aeropêndulo foi usado um Conjunto (suporte/motor/hélice) para drones fpv racing quadcopter. A Figura 14 ilustra o componente em questão.

Figura 14 – Conjunto (suporte motor cw/ccw hélice).



Fonte: elaborado pelo autor (2023).

#### 2.2.3.6 Componentes eletrônicos (resistivos e capacitivos)

Por fim, para que o sinal do sensor (Potenciômetro) seja de boa qualidade, foi usado um filtro RC série, dessa forma, empregou-se um capacitor de um resistor na implementar do filtro. A Figura 15a corresponde aos componentes resistivos e a Figura 15b aos componentes capacitivos.

Figura 15 – Componentes eletrônicos.

(a) Resistores.



Fonte: elaborado pelo autor (2023).

(b) Capacitores.



Fonte: elaborado pelo autor (2023).

## 2.2.4 Montagem do Protótipo

### 2.2.4.1 Parte Física

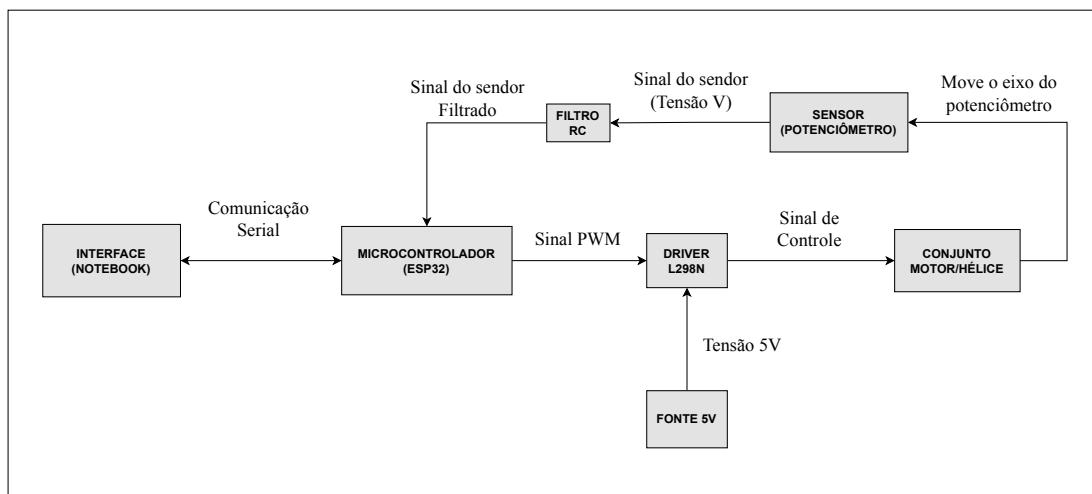
O protótipo foi concebido visando a desmontagem da estrutura, permitindo um transporte mais conveniente. A construção física compreende três componentes com pontos de conexão estratégicos. Ademais, o braço do Aeropêndulo pode ser facilmente desacoplado no ponto de pivô, onde se conecta ao eixo do potenciômetro.

Para montar a estrutura, basta acoplar suas partes. Com isso, a componente estrutural estará pronta para uso. O resultado final do sistema ficou notavelmente estável, com a estrutura demonstrando rigidez suficiente para evitar vibrações indesejadas no braço do Aeropêndulo durante o acionamento.

### 2.2.4.2 Parte Elétrica

A Figura 16 ilustra a dinâmica do fluxo de interligação do sistema elétrico do Aeropêndulo. Notavelmente, o microcontrolador desempenha um papel central ao gerar o sinal de controle PWM, realiza a leitura do sinal filtrado proveniente do sensor (Potenciômetro) e estabelecer comunicação com o computador através da interface serial. Adicionalmente, o driver é alimentado por uma fonte de tensão contínua de 5V, o qual amplifica e aplica o sinal de controle ao motor CC Série. Esta ação, por sua vez, resulta em uma variação angular no braço do Aeropêndulo por conta do empuxo gerado pelas hélices, impactando diretamente o estado do sensor (Potenciômetro) e sua saída correspondente.

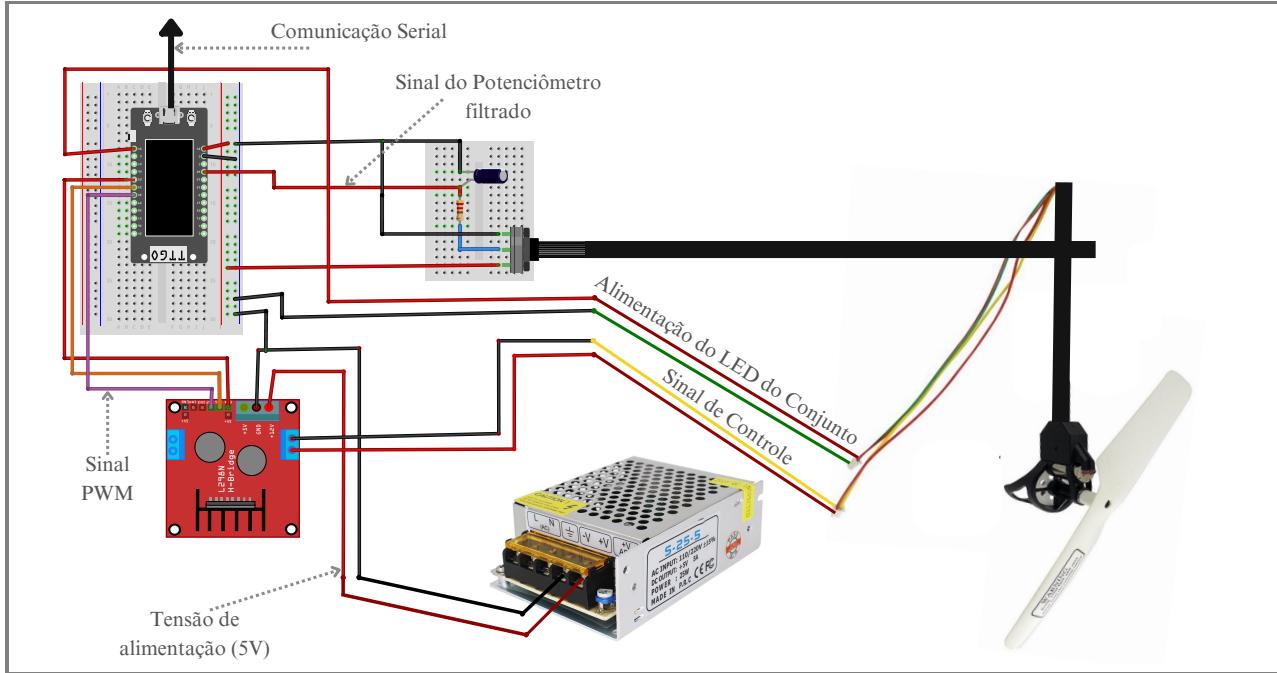
Figura 16 – Diagrama de comunicação do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

O esquema de ligação dos componentes do sistema elétrico é exemplificado na Figura 17.

Figura 17 – Esquema de conexões elétricas do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

## 2.3 Desenvolvimento dos Softwares

Para permitir a interação com o protótipo um conjunto de software foi desenvolvido, permitindo aos usuários aplicar diferentes conceitos de sistema de controle e obter uma visualização em tempo real dos estados do sistema através desses softwares, sendo eles, o firmware do microcontrolador, simulador 3D que usa o conceito de Gêmeo Digital, uma interface gráfica com um conjunto de funcionalidades capaz de manipular o sinal de referência e plotar os gráficos dos estados do sistema.

### 2.3.1 Linguagens Python, C e C++

Para o desenvolvimento dos softwares destinados à visualização de dados e ao simulador, a linguagem de programação escolhida foi o Python. Essa seleção se deu devido à natureza versátil da linguagem, que permite um desenvolvimento ágil de softwares para uma ampla gama de finalidades. Em paralelo, para a programação do microcontrolador, optou-se pelo uso das linguagens C/C++. Essa escolha se fundamenta na ampla adoção dessas linguagens na programação de sistemas embarcados, garantindo um ambiente propício para a eficaz implementação no microcontrolador.

A linguagem Python é amplamente reconhecida como uma linguagem de programação de alta qualidade, interpretada e de propósito geral. Ela desfruta de uma imensa popularidade em todo o mundo e é empregada em diversas aplicações, abrangendo campos como computação

científica, ciência de dados, engenharia de software e inteligência artificial. Python se destaca por ser uma linguagem de programação de fácil aprendizado e utilização, mesmo para indivíduos com pouca experiência em codificação. Sua sintaxe clara e concisa contribui para a legibilidade e manutenibilidade do código.

Por essas razões, Python emerge como a escolha ideal para conduzir trabalhos de pesquisa e desenvolvimento. É uma ferramenta incrivelmente versátil e poderosa, capaz de lidar com uma extensa variedade de tarefas, desde a análise de dados até a criação de aplicações complexas.

Já C e C++ são linguagens de programação que se concentram na eficiência e na portabilidade. Elas são amplamente utilizadas no desenvolvimento de sistemas operacionais, drivers de dispositivos, compiladores e outros softwares críticos. C foi criada em 1972 por Dennis Ritchie para o desenvolvimento do sistema operacional Unix. C++ foi criada em 1983 por Bjarne Stroustrup como uma extensão de C para adicionar suporte à programação orientada a objetos.

C e C++ são linguagens de programação poderosas e flexíveis, mas também podem ser complexas e difíceis de aprender. Elas são recomendadas para desenvolvedores que precisam de um alto nível de controle sobre o desempenho e a portabilidade de seu código.

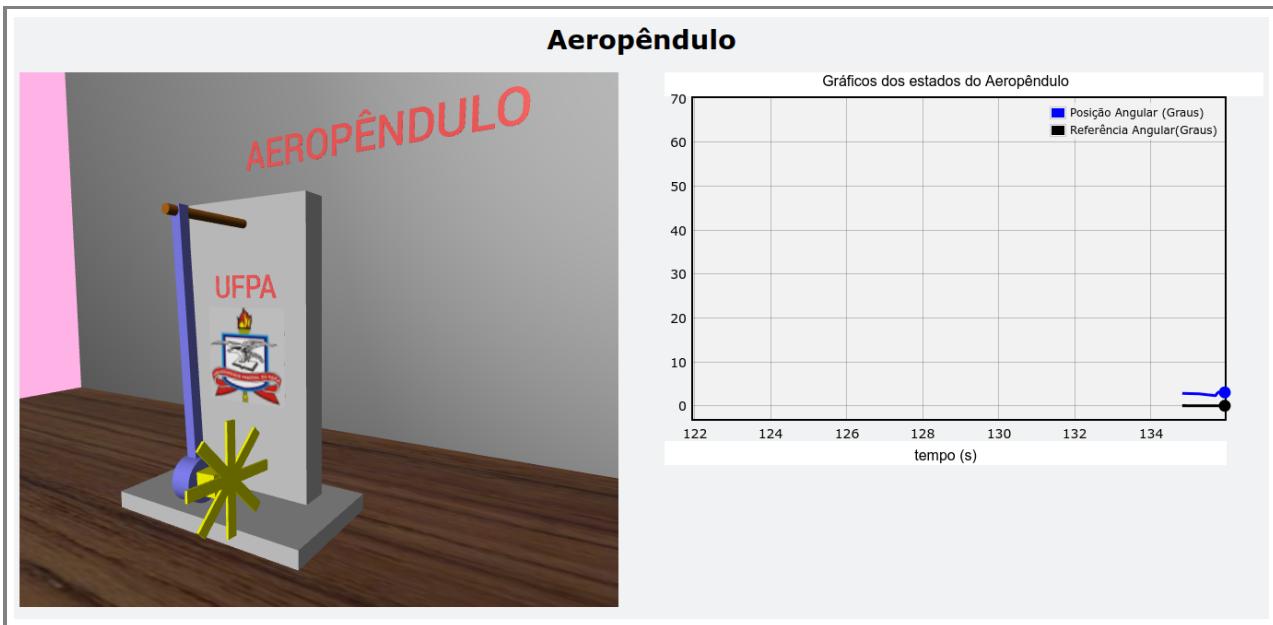
### 2.3.2 Gêmeo Digital

Nesta seção, será elaborado o Gêmeo Digital do Aeropêndulo. O Gêmeo Digital representa um sistema gráfico computacional que reproduz em tempo real a dinâmica do protótipo. Isso permite a virtualização do Aeropêndulo, possibilitando a observação da mesma dinâmica do sistema físico, agora em um ambiente gráfico computacional 3D. Essa virtualização é viabilizada pela obtenção do estado atual do braço do Aeropêndulo por meio de comunicação serial.

Para o desenvolvimento do Gêmeo Digital, utilizou-se a biblioteca VPython, essa biblioteca possui um conjunto de funções que permite criar objetos 3D capazes de realizar movimentos rotacionais de translacionais, além disso, é possível plotar gráficos em tempo real. A Figura 19 ilustra a interface do simulador já finalizado, pode-se observar que a interface é composta de duas partes principais, uma que implementa o ambiente 3D com o Aeropêndulo e a outra gera os gráficos do sinal de referência e de saída.

Com a finalidade de atualizar os estados dos gráficos e a posição angular do gêmeo digital em tempo real a aplicação se comunica com a interface gráfica, subseção 2.3.3, com isso é possível realizar a atualização dos estados do gráfico assim como do gêmeo digital tendo como entrada os sinais dos estados do protótipo.

Figura 18 – Gêmeo Digital - Simulador com VPython.



Fonte: elaborado pelo autor (2023).

### 2.3.2.1 Biblioteca VPython

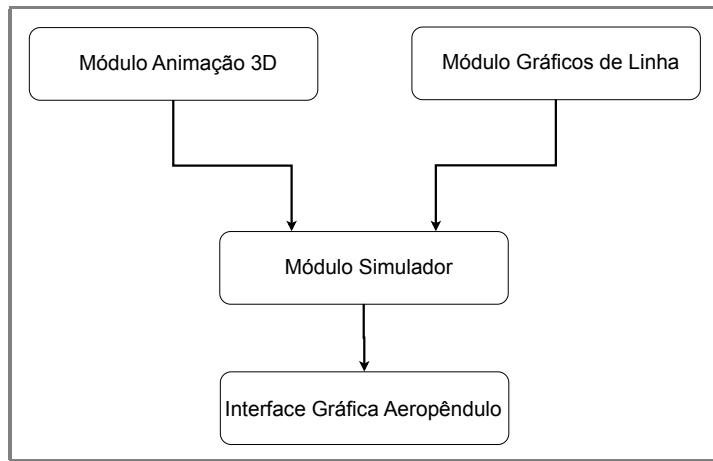
Conforme o site oficial ([VPYTHON.ORG, 2023](https://www.vpython.org)), "O VPython, por ser desenvolvido com a linguagem Python, é uma ferramenta versátil para a criação de animações 3D navegáveis. Ele é fácil de aprender e usar, mesmo para pessoas com pouca experiência em programação. No entanto, ele também oferece uma ampla gama de recursos para programadores e pesquisadores experientes."

Ao criar uma simulação com VPython e executá-la, o simulador será renderizado no navegador de internet padrão do sistema operacional.

### 2.3.2.2 Arquitetura do Gêmeo Digital

Para implementar o simulador a arquitetura do sistema consiste de um módulo para gerar os gráficos de linha e outro para desenhar o simulador 3D, existe um módulo que integra e atualiza os estados dos gráficos de linha e do simulador 3D, a Figura 19 mostra como a estrutura do simulador foi pensada, o último bloco é a interface gráfica, subseção 2.3.3, responsável por obter os estados do sistema real, com isso, é possível usar a velocidade angular real do protótipo como entrada para o gêmeo digital, isso faz com que a dinâmica do protótipo seja reproduzida no gêmeo digital, além disso, é possível obter os sinais de referência e de saída e atualizar os gráficos que compõem a interface do simulador.

Figura 19 – Diagrama da arquitetura do Gêmeo Digital.



Fonte: elaborado pelo autor (2023).

O **Módulo Simulador**, código abaixo, recebe como parâmetro o **Módulo Animação 3D** e o **Módulo Gráficos de Linha**, a partir disso é implementada uma Classe Python usando a ideia de programação orientada a objetos para realizar a construção tanto da parte gráfica quanto do simulador 2D.

```

import vpython as vp

# Módulos do Gêmeo Digital desenvolvidos
from .interfaces.graficos_aeropendulo import GraficosInterface
from .interfaces.animacao_aeropendulo import AnimacaoAeropenduloInterface
from .interfaces.simulador import SimuladorInterface


class Simulador(SimuladorInterface):

    def __init__(self, graficos: GraficosInterface,
                 animacao_aeropendulo: AnimacaoAeropenduloInterface) -> None:
        self.t = 0
        self.t_ant = 0
        self.ts = 0
        self.theta_rad = 0
        self.theta_rad_ant = 0
        self.dtheta_rad = 0
        # Instanciando um objeto AeropenduloAeropendulo()
        self.animacao_aeropendulo = animacao_aeropendulo
  
```

```
# Instanciando um objeto para plotagem dos graficos
# dinamicos dos estados do Aeropendulo
self.g = graficos
self.graf, self.plot1, self.plot2 = self.g.graficos() # noqa

def grau2rad(self, graus):
    return (graus)*(vp.pi/180.0)

def rotate(self, angle) -> None:
    self.valor_angle = self.grau2rad(angle)
    self.animacao_aeropendulo.aeropendulo.rotate(
        axis=vp.vec(0, 0, 1),
        angle=self.valor_angle,
        origin=vp.vec(0, 5.2, 0))
    self.animacao_aeropendulo.set_posicao_helice(self.valor_angle)

def atualizar_estados(self, t, theta, ref):
    self.t = t
    self.ts = self.t - self.t_ant
    self.theta_rad = self.grau2rad(theta)
    try:
        self.dtheta_rad = (
            self.theta_rad - self.theta_rad_ant
        )/self.ts
    except Exception as exception:
        print(exception)

    # Atualiza o angulo do Aeropendulo
    self.animacao_aeropendulo.aeropendulo.rotate(
        axis=vp.vec(0, 0, 1),
        angle=self.dtheta_rad*self.ts,
        origin=vp.vec(0, 5.2, 0))

    # Anima o da dinamica da Helice
    self.animacao_aeropendulo.update_helice(self.dtheta_rad, self.ts)

    # print(x[1] + interface.valor_angle)
    # Gráfico do angulo .
    self.plot1.plot(t, theta)
    # Gráfico do sinal de referência
    self.plot2.plot(t, ref)

    self.t_ant = t
    self.theta_rad_ant = self.theta_rad
```

Para atualizar a dinâmica do braço do Aeropêndulo do Gêmeo Digital, a classe **Simulador** é importada no outro software, seção 2.3.3 que implementa a comunicação com o protótipo e obtém os estados do sistema real, assim é possível atualizar o Gêmeo Digital com dados reais do braço do Aeropêndulo.

Logo, o código desenvolvido para implementar o Gêmeo Digital deve ser importado no código que implementa a Interface Gráfica de Usuário, seção 2.3.3.

### 2.3.3 Interface Gráfica para Configuração, visualização e aquisição de dados da Planta

A arquitetura da interface gráfica possui o estrutura mostrada na Figura ??.

O código abaixo importa todas os Módulos Python desenvolvidos tanto para a implementação do Interface Gráfica de Usuário quanto para o Gêmeo Digital, a partir disso foi desenvolvido um algorítimo que estrutura corretamente o funcionamento dos dois softwares e permite que eles se comuniquem.

```
import argparse
from src_interface import InterfaceAeropendulo
from src_interface.graficos_sinais import GraficosSinais

from simulador_aeropendulo.simulador import Simulador
from simulador_aeropendulo.graficos_aeropendulo import Graficos
from simulador_aeropendulo import AnimacaoAeropendulo


class RunInterface:

    def __init__(self) -> None:
        simular = self.get_args()
        self.runinterface(simular)

    def get_args(self) -> bool:
        parser = argparse.ArgumentParser()
        parser.add_argument("-simular", "--Output",
                            help="""Para habilitar o simulador, use a
                                    syntax:
                                    python rungui.py -simular sim""")
        args = parser.parse_args()
        if args.Output == "sim":
            return True
```

```

    else:
        return False

def runinterface(self, simular):
    if simular:
        simulador = Simulador(Graficos(), AnimacaoAeropendulo())
    else:
        simulador = None
    InterfaceAeropendulo(GraficosSinais, simulador, baud_rate=115200,
                          amostras=80.0, tela_fixa=True)

if __name__ == "__main__":
    RunInterface()

```

Para roda a Interface Gráfica de Usuário juntamente com o Gêmeo Digital usa-se o seguinte comando no terminal estando dentro da pasta a partir do terminal:

```
python rungui.py -simular sim
```

Para roda apenas a Interface Gráfica de Usuário usa-se o seguinte comando no terminal:

```
python rungui.py
```

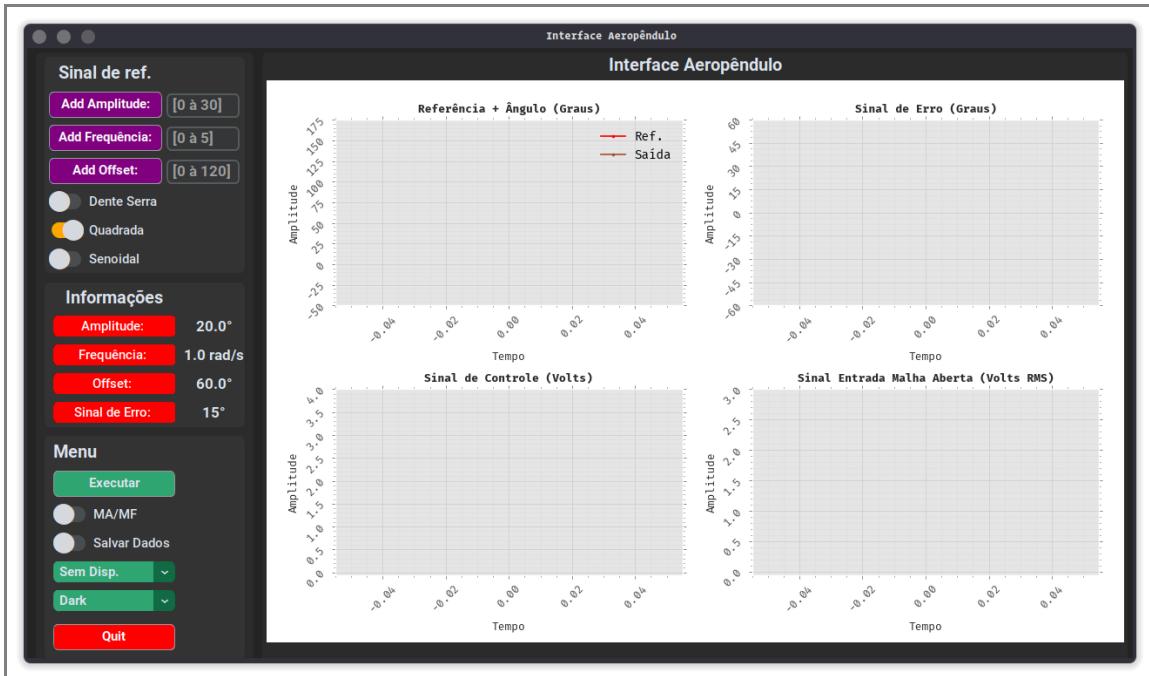
A interface gráfica consiste de um menu com diferentes opções de controle e visualização dos sinais do sistema e um conjunto de gráficos dinâmicos para visualizar os estados do sistema em tempo real. A Figura 20 mostra a interface do sistema. No entanto, além da parte visual existe a parte de aquisição e processamento de dados, em TI essa parte é chamada de Back-end, o projeto possui uma classe para realizar essa parte de aquisição e tratamento de dados, além de uma classe para detectar automaticamente o microcontrolador conectado a porta USB.

Para ser possível realizar os ensaios, modificação de parâmetros do sinal de referência e coletas de dados é preciso que o microcontrolador execute o firmware, subseção 2.3.4, desenvolvido especificamente para esse propósito, dessa forma, a interface gráfica, Figura 20, conseguirá realizar o pre-processamentos dos dados e plotar os sinais nos gráficos e demais atualizações nos softwares.

### 2.3.3.1 Biblioteca CustomTkinter e Matplotlib

Para o desenvolvimento da parte gráfica do software foi usado as bibliotecas CustomTkinter e Matplotlib, onde a biblioteca CustomTkinter implementa a estrutura de tela, botões, entradas de texto da aplicação e os módulos *Matplotlib.pyplot* e *Matplotlib.animation* possibilitam gerar gráficos em tempo real em conjunto com CustomTkinter.

Figura 20 – Interface Gráfica.



Fonte: elaborado pelo autor (2023).

Conforme ([SCKIMANSKY TOM, 2023](#)) explica no site oficial da biblioteca, "CustomTkinter é uma biblioteca de UI de desktop python baseada em Tkinter, que fornece widgets de aparência moderna e totalmente personalizáveis. Com CustomTkinter você terá uma aparência consistente em todas as plataformas de desktop (Windows, macOS, Linux)."

De acordo com ([MATPLOTLIB DEVELOPMENT TEAM, 2023](#)), "Matplotlib é uma biblioteca abrangente para criar visualizações estáticas, animadas e interativas em Python. Matplotlib torna as coisas simples fáceis e as difíceis possíveis."

### 2.3.3.2 Biblioteca PySerial, NumPy e Pandas

Já para implementar o Back-end foram utilizadas as bibliotecas PySerial, NumPy e Pandas. Sendo que a biblioteca PySerial tem por finalidade realizar a comunicação entre o computador e o microcontrolador usando o protocolo serial via entrada USB, já o NumPy teve sua aplicação no processo de criação das matrizes contendo os dados lidos pela PySerial, por fim, para salvar os dados de ensaio foi usada a biblioteca Pandas, sendo salvos no formato CSV.

Conforme ([LIECHTI CHRIS, 2023](#)), "A biblioteca PySerial possibilita a comunicação serial entre o computador e o microcontrolador usando conexão USB. Este módulo encapsula o acesso à porta serial. Ele fornece backends para Python rodando em Windows, OSX, Linux, BSD (possivelmente qualquer sistema compatível com POSIX) e IronPython. O módulo denominado "serial" seleciona automaticamente o backend apropriado."

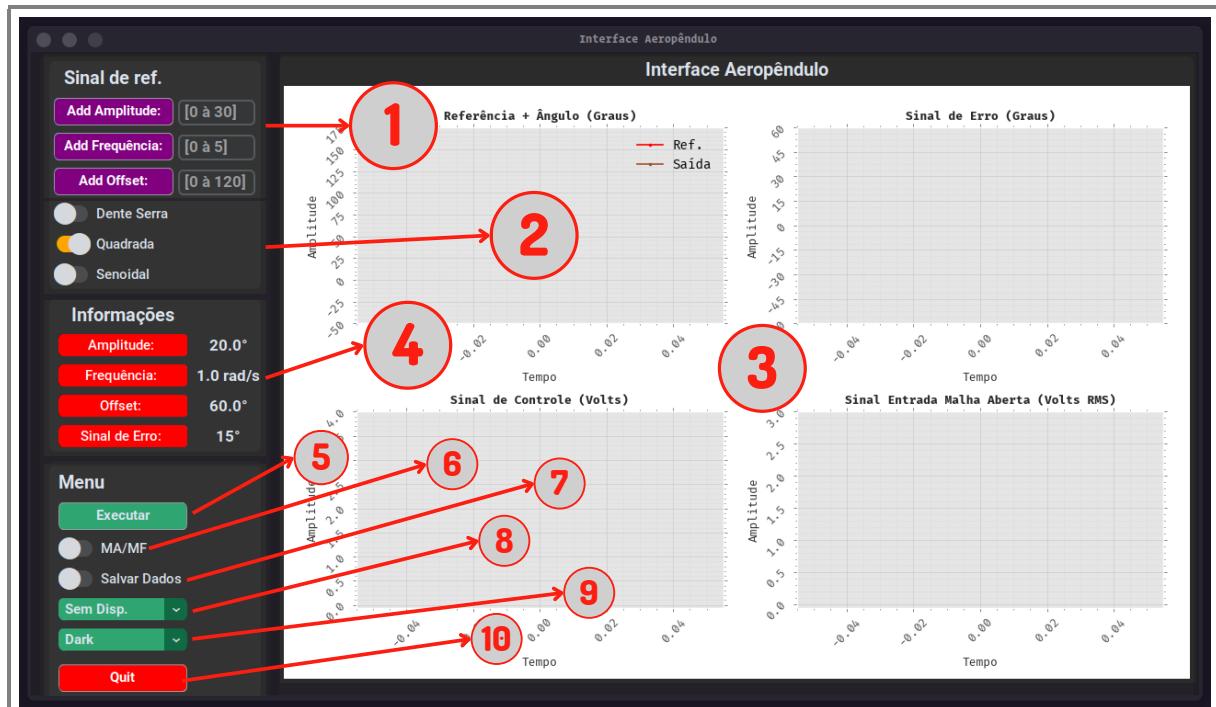
De acordo com (NETO ANSELMO, 2023), "O NumPy, uma biblioteca essencial para Python, oferece recursos robustos para realizar cálculos numéricos, sendo seu elemento central o ndarray, também conhecido como tensor. O ndarray se destaca por sua homogeneidade, exigindo que todos os elementos pertençam ao mesmo tipo, em contraste com as listas Python."

Segundo (PANDAS DEVELOPMENT TEAM, 2023), site oficial da ferramenta, "pandas é uma ferramenta de análise e manipulação de dados de código aberto rápida, poderosa, flexível e fácil de usar, construído sobre a linguagem de programação Python."

### 2.3.3.3 Descrição das partes da interface gráfica

Nessa subseção será detalhado as partes do software, a Figura 21 está enumerando as partes e suas descrições estão logo abaixo.

Figura 21 – Partes da Interface Gráfica.



Fonte: elaborado pelo autor (2023).

#### *Item 1: Sinal de Referência*

Essa parte do software configura os sinais de referência aplicado ao sistema em malha fechada.

- **Add Amplitude:** Configura a amplitude do sinal aplicado a entrada do sistema;
- **Add Frequência:** Configura a frequência do sinal aplicado a entrada do sistema;
- **Add Offset:** Configura a amplitude do offset, ponto de equilíbrio, aplicado a entrada do sistema.

*Item 2: Seleção do Sinal de referência*

- **Onda Serra:** Ao selecionar essa opção o sinal aplicado ao sistema será um dente de serra com as configurações de amplitude, frequência e offset conforme configurado nos três primeiros sub itens;
- **Quadrada:** Ao selecionar essa opção o sinal aplicado ao sistema será uma onda quadrada com as configurações de amplitude, frequência e offset conforme configurado nos três primeiros sub itens.;
- **Senoidal:** Ao selecionar essa opção o sinal aplicado ao sistema será uma onda senoidal com as configurações de amplitude, frequência e offset conforme configurado nos três primeiros sub itens.

*Item 3: Gráficos*

- **Gráficos:** Nesse item que são plotados os gráficos dos estados do sistema em tempo real.

*Item 4: Informações*

- **Amplitude:** Mostra a valor da amplitude do sinal de referência;
- **Frequência:** Mostra a valor da frequência do sinal de referência;
- **Offset:** Mostra a valor de offset aplicado ao aeropêndulo, esse valor é adicionado ao sinal de referência, dessa forma o sinal de referência varia em torno de um ponto de equilíbrio;
- **Sinal de Erro:** Mostra a valor do sinal de erro em tempo real.

*Item 5: Menu*

- **Executar:** Inicializa o ensaio, para isso é preciso que o microcontrolador esteja conectado ao computador.

*Item 6: Menu*

- **MA/MF:** Comuta entre o sistema em malha aberta e malha fechada.

*Item 7: Menu*

- **Salvar Dados:** Quando está ativado os dados são salvos em um arquivo CSV com data, hora, minuto e segundo do instante do salvamento, isso facilita que diferentes ensaios sejam realizados sem que os dados do anterior seja sobreescrito.

*Item 8: Menu*

- **Selecionar Dispositivo:** A aplicação detecta todos os microcontroladores conectados no computador e permite que o usuário realize a seleção do dispositivo desejado para realizar o ensaio.

*Item 9: Menu*

- **Selecionar Tema:** O software possui tema escuro e claro e essa opção permite que o usuário selecione entre essas duas opções.

*Item 10: Menu*

- **Quit:** Botão responsável por fechar a aplicação.

### 2.3.4 Firmware do microcontrolador

O firmwawe para o aeropêndulo foi desenvolvido a partir de bibliotecas, cada biblioteca tem sua funcionalidade, com isso o projeto se torna mais flexível, podendo ser atualizado e incrementado a medida que o projeto se torne mais robusto.

Para desenvolver o firmware, utilizamos a ferramenta de plataforma cruzada PlatformIO. Essa ferramenta foi criada com o propósito de centralizar o desenvolvimento de firmware, possibilitando a programação para diversos microcontroladores, como o ESP32, a Família do Arduino, STM32 e muitos outros. Isso torna o projeto ainda mais flexível e versátil. A arquitetura é descrita na Figura 22.

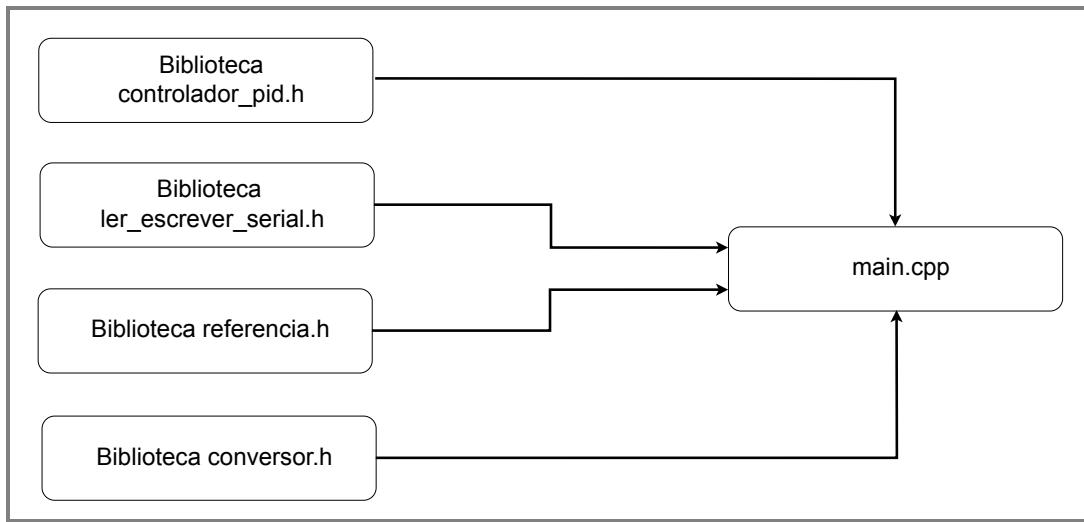
#### 2.3.4.1 Biblioteca ler\_escrever\_serial

Para realizar a configuração dos parâmetros do sinal de referência em tempo real e o envio dos sinais do sistema via porta serial foi desenvolvido a biblioteca ler\_escrever\_serial, assim, a parte de recebimento e envio de dados do sistema fica centralizada permitindo sua melhoria e modificação sem interferir na lógica do código principal "main.cpp".

#### 2.3.4.2 Biblioteca referencia

O sistema em malha fechada possui uma entrada de referência que o controlador tem por finalidade rastrear, para tornar o projeto do firmware mais legível criou-se o módulo referencia.h em que implementa inicialmente três sinais de referência com a possibilidade de configurar os parâmetros de Amplitude, frequência e offset, sendo esses sinais Onda Quadrada, Onda Senoidal e Onda Dente de Serra. Caso, observe-se a necessidade de outros sinais, basta adicionar a biblioteca.

Figura 22 – Arquitetura do Firmware do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

#### 2.3.4.3 Biblioteca conversor

Essa biblioteca possui uma classe cpp com métodos de conversões de diferentes grandezas, a primeira conversão é do sinal de um potenciômetro para ângulo, o segundo método converte o sinal de controle para ciclos PWM, o terceiro método converte de grau para radiano e o quarto de radiano para grau.

#### 2.3.4.4 Biblioteca controlador\_pid

Essa biblioteca implementa uma classe cpp para o controlador PID para ser empregado no protótipo quando selecionado a opção em malha fechada na interface gráfica.

#### 2.3.4.5 Arquivo Principal main.cpp

A ferramenta PlatformIO possui uma estrutura que permite a criação de bibliotecas e um arquivo main.cpp que implementa a lógica do algoritmo, com isso é possível incluir as bibliotecas desenvolvidas e criar o algoritmo para o que se deseja.

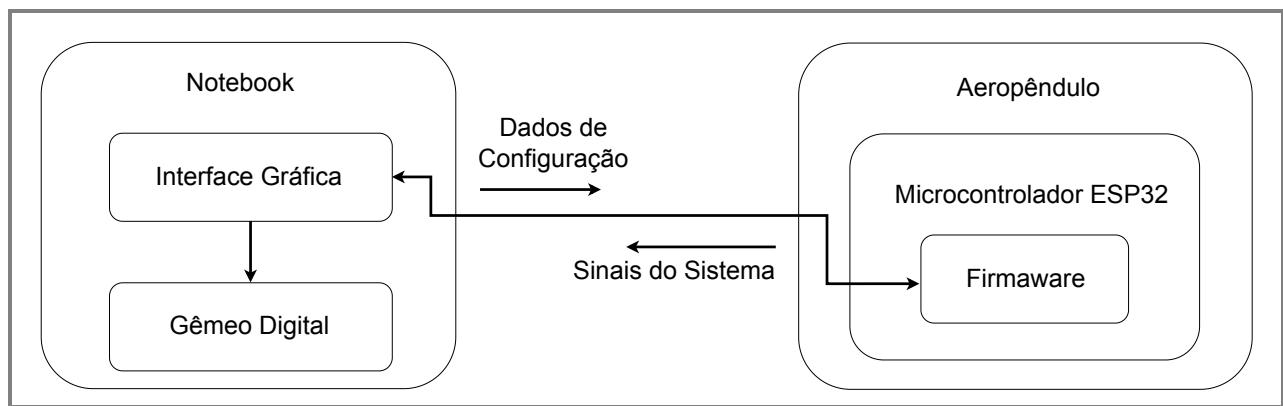
Ou seja, o firmware para o projeto possui várias funcionalidades que passam pela implementação do sinal de referência, leitura do sensor potenciômetro, envio e recebimento de dados via porta serial e implementação do controlador.

Por fim, com o firmware finalizado, o envio para o ESP32 é realizado usando o PlatrmIO que concretiza a compilação e escrita no microcontrolador via porta serial.

## 2.4 Fluxograma do Laboratório Virtual

Dado o caráter multifacetado deste projeto, a compreensão das interações entre seus diversos subsistemas pode apresentar desafios significativos. Com o propósito de simplificar e aprofundar esse processo de compreensão, a Figura 23 oferece uma representação visual por meio de um diagrama de blocos que esclarece as dinâmicas entre os distintos subsistemas desenvolvidos. Essa abordagem contribui para facilitar o entendimento holístico do sistema na totalidade, proporcionando um recurso valioso para pesquisadores futuros que desejam explorar e aprimorar este ecossistema.

Figura 23 – Diagrama do Laboratório Virtual.



Fonte: elaborado pelo autor (2023).



# RESULTADOS E DISCUSSÕES

---

---

## 3.1 Desenvolvimento do Protótipo e Softwares

Os sistemas propostos tanto físico quanto softwares foram desenvolvidos de forma exitosa, os procedimentos para a implementação se deu a partir de uma pesquisa bibliográfica e criativa, tendo como resultado o protótipo, Figura 7, e os softwares, Figuras 19, 19 e 22.

## 3.2 Identificação de sistema aplicado ao Aeropêndulo

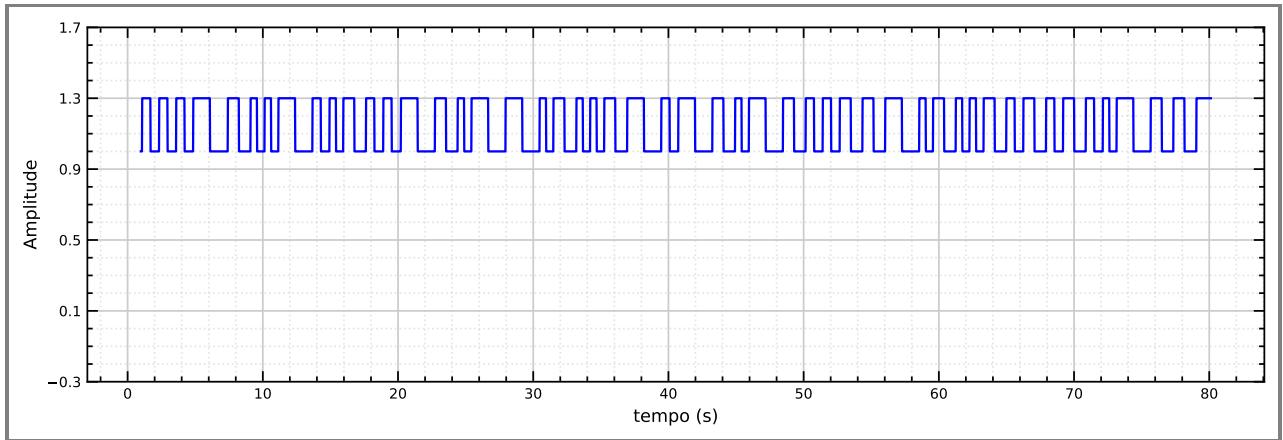
Com os subsistemas desenvolvidos, foram realizados alguns testes para validá-los, assim, o primeiro teste aplicado ao conjunto se tratou da identificação de sistemas. Após a identificação realizou-se uma simulação de modo a comparar o sinal de saída real com o simulado, visando visualizar a dinâmica do sistema identificado e poder comparar com o sistema real.

A partir do ecossistema desenvolvido para o projeto é possível usá-lo para estudar diferentes aspectos na área de sistemas de controle, uma delas é a identificação de sistemas, para esse trabalho o método em questão foi usado para avaliar o correto funcionamento projeto. Desse forma, o trabalho ([KLARISSA, 2023](#)) foi usando como base teórica para implementar o método.

### 3.2.0.1 Aplicação do Sinal PRBS na Entrada do Sistema

O sinal PRBS aplicado a entrada do sistema em malha aberta foi gerado no microcontrolador e convertido em sinal PWM para ser aplicado a entrada do Driver L298N, subseção 2.2.3.4, seus parâmetros foram definidos com uma amplitude de 0,3V, frequência fundamental de 0,4Hz e período de amostragem de 0,02 segundos, além disso, aplicou-se um offset de 1V (Ponto de operação), com isso, obteve-se o sinal PRBS de entrada mostrado na Figura 22.

Figura 24 – Sinal PRBS de entrada.

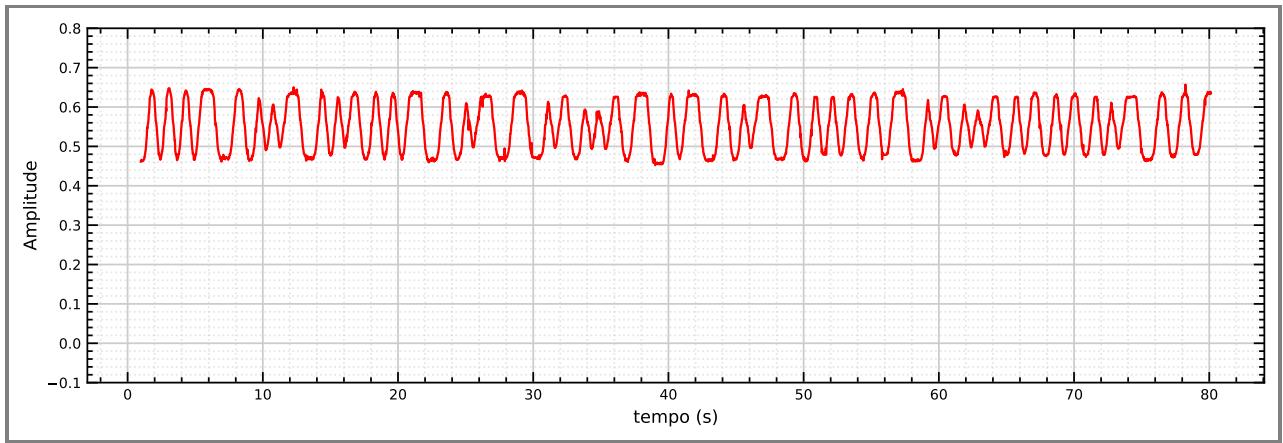


Fonte: elaborado pelo autor (2023).

### 3.2.0.2 Sinal de Saída Gerado a Partir do Sinal PRBS de Entrada

Para a obtenção do sinal de saída, que representa o ângulo do braço do aeropêndulo, foi empregado um método no qual um sinal PRBS foi aplicado à entrada do sistema. Para medir o ângulo, sinal de saída, utilizou-se um potenciômetro como sensor, cujas extremidades foram conectadas ao 3.3V e GND do ESP32. O terceiro fio do potenciômetro foi conectado a uma porta analógica do ESP32, permitindo, assim, a conversão da variação de tensão em valores angulares. A Figura 25 exibe o sinal de saída de um ensaio.

Figura 25 – Sinal PRBS de saída.



Fonte: elaborado pelo autor (2023).

### 3.2.1 Identificação Usando Mínimos Quadrados

Para realizar a identificação de sistemas deve-se retirar os offsets tanto do sinal de entrada quanto para o de saída, com essa etapa realizada, deve-se separar dois conjuntos de dados para

identificação e validação do modelo proposto, tanto para o sinal de entrada quanto para o de saída. Foi usando 60% dos dados para identificação de 40% para validação.

### 3.2.1.1 Passos para realização da identificação

*Passo 1:* Obter dados Entrada/Saída do sistema;

*Passo 2:* Dividir dados entre identificação e validação;

*Passo 3:* Definir a Função de Transferência;

*Passo 4:* Definir a matriz de regressão;

*Passo 5:* Encontrar o vetor de coeficientes pela formulação de mínimos quadrados;

*Passo 6:* Substituir os coeficientes na função de transferência escolhida anteriormente;

*Passo 7:* Validar o modelo do sistema a partir de dados reais.

### 3.2.1.2 Identificação Usando Função de Transferência Discreta de Segunda, Quinta e Décima Ordem

Para realizar o cálculo numérico usando a base de dados do ensaio foi usando a linguagem de programação Python, que possui um conjunto de bibliotecas que facilita o implementação dos cálculos. Após obter os dados de ensaio, os passos seguintes foram desenvolvidos em um Script Python, sendo o primeiro passo a importação das bibliotecas, como mostra a célula de código abaixo.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import control as ct
```

No segundo passo foi feito o carregamento dos dados de ensaio PRBS e realizada a separação entre dados de identificação e validação, para isso foi usado a biblioteca Pandas em conjunto com a biblioteca NumPy que possibilita carregar dados de diferentes formatos e manipulá-los, os dados de ensaio foram salvos em formato CSV (Comma-separated values), o código para realizar o carregamento dos dados e a separação é mostrado na célula de código abaixo.

```
file = "caminho completo dos dados/arquivo_9_9_2023_13_33_24.csv"
dados_malha_aberta_prbs = pd.read_csv(file, header=None, sep=',').values
dados_malha_aberta_prbs[0][0] = 0.0
dados_malha_aberta_prbs

# Variaveis contendo os dados de tempo, entrada e saida
tempo = np.array(dados_malha_aberta_prbs[:,7])
```

```

sinal_prbs_entrada = np.array(dados_malha_aberta_prbs[:, 6])
sinal_saida = np.array(dados_malha_aberta_prbs[:, 2])

# Convertendo o sinal de Graus para Radianos
sinal_saida = np.squeeze(np.deg2rad(sinal_saida))
tempo = tempo - min(tempo)

```

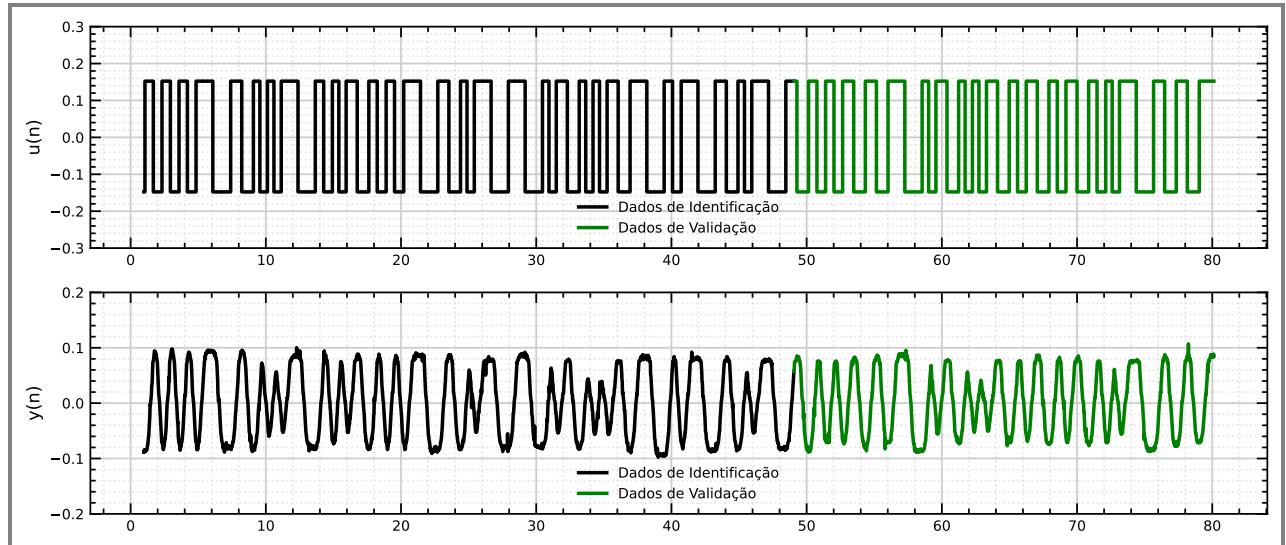
Além disso, é necessário remover as primeiras amostras por conter o transitório do sistema e retirar os offsets dos sinais de entrada e saída, o que é feito na célula de código abaixo. A figura 26 exibe os gráficos dos sinais de entrada/saída, podemos observar que os sinais estão centralizados em zero e não possui transitório no início do sinal.

```

u1 = sinal_prbs_entrada[50:] - np.mean(sinal_prbs_entrada[50:])
yout = sinal_saida[50:] - np.mean((sinal_saida[50:]))
t = tempo[50:]

```

Figura 26 – Dados de identificação e validação do modelo.



Fonte: elaborado pelo autor (2023).

Com os dados de ensaio pré-processados e prontos para aplicar a técnica de identificação, foi definida uma função de transferência discreta, essa função tem a finalidade de aproximar um modelo que satisfaça a dinâmica do aeropêndulo. Para um primeiro teste foi usada uma função de segunda ordem, como mostra a equação 3.1.

$$H(z) = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (3.1)$$

Para aplicar o método dos mínimos quadrados e obter os coeficientes da função de transferência de segunda ordem precisa-se ter as matrizes de regressão tanto para os dados de entrada quanto para os de saída, o código abaixo cria o algoritmo necessário para se obter as matrizes a partir dos dados do ensaio.

```
# Matriz de regressao:
nb = 3; na = 2
ni = np.arange(na, Ni + na)
M = np.zeros((Ni, na + nb))

# Para regressores de y:
for l in np.arange(0, na):
    M[:, l] = yout[ni - l - 1]

# Para regressores de u:
for l in np.arange(0, nb):
    M[:, na + l] = u1[ni - l]

print(M.shape)
```

Com as matrizes de regressores definidas, foi realizada a aplicação do método dos mínimos quadrados, o que é exemplificado na célula de código abaixo, a biblioteca NumPy possibilita cálculos de álgebra matricial de forma rápida e simples.

```
# Minimos quadrados
thetaA = np.linalg.inv(M.T @ M) @ M.T @ yout[ni]
thetaA
```

Para a função de transferência de segunda ordem escolhida, equação 3.1, em que possui dois coeficientes no numerador e 3 no denominador, dessa forma, os coeficientes obtidos ao aplicar o método de mínimos quadrados retorna em suas duas primeiras posições os coeficientes do numerador e o restante são os coeficientes do denominador, a partir dos coeficientes obtidos, pode-se definir a função de transferência discreta do sistema, isso é feito no Python usando a biblioteca Python-Control que permite realizar ensaios simulados a partir da função de transferência.

```
a1, a2 = thetaA[:2]
b0, b1, b2 = thetaA[2:]

Ba = [b0, b1, b2]
Aa = [1, -a1, -a2]

Gz = ct.tf(Ba, Aa, Ts)
```

Função de transferência discreta, 3.2, com os coeficientes obtidos a partir do método de mínimos quadrados.

$$\frac{-0.002602z^2 + 0.004962z + 0.0163}{z^2 - 1.176z + 0.1849} \quad dt = 0.019 \quad (3.2)$$

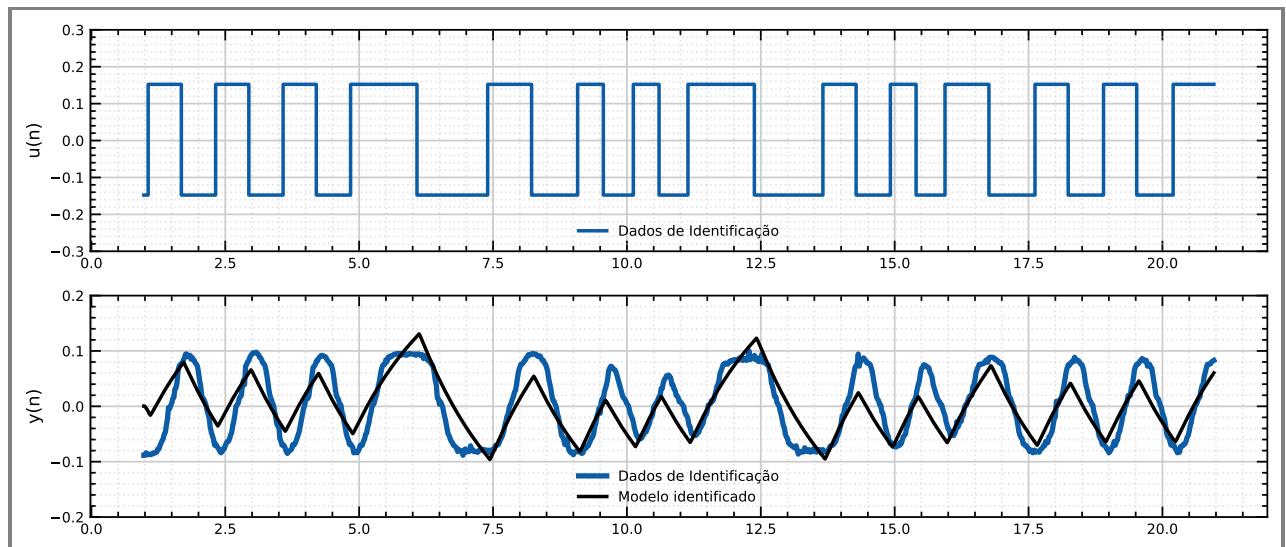
Para validar o modelo encontrado é preciso testar a função de transferência discreta usando os dados de entrada do ensaio real, com isso obtém-se os dados de saída da simulação, esse dado pode ser comparado com a saída do sistema real, caso ambos os gráficos aproximem suas dinâmicas pode-se concluir que o modelo corresponde com o sistema real, caso contrário é preciso definir uma nova função de transferência e realizar novamente o teste de validação.

A biblioteca Python-control possibilita simular a dinâmica de um sistema a partir da sua função de transferência discreta, podendo receber como argumento um vetor de entradas, como mostrado a célula de código abaixo.

```
_ , yp = ct.forced_response(Gz, U=u1)
```

Ao plotar os gráficos de saída real e simulada tem-se como resultado a Figura 27, é possível observar que a saída da simulação não corresponde de forma aceitável com a dinâmica do sinal de saída do sistema real, ou seja, o modelo não foi robusto o suficiente para identificar o sistema em questão.

Figura 27 – Validação do modelo de segundo grau.



Fonte: elaborado pelo autor (2023).

A partir de uma análise qualitativa, pode-se supor que o grau da função de transferência discreta escolhida não consegue aproximar a dinâmica real do aeropêndulo, dessa forma, foi

realizado alguns testes e identificado que uma função de transferência discreta de décima ordem, equação 3.3, aproxima de forma aceitável o sistema real.

$$H(z) = \frac{b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4} + a_5 z^{-5} + a_6 z^{-6} + a_7 z^{-7} + a_8 z^{-8} + a_9 z^{-9} + a_{10} z^{-10}} \quad (3.3)$$

Assim, foi ajustado o algorítimo para identificar o sistema a partir da estrutura da equação 3.3, o código abaixo gera as matrizes tanto para os regressores de entrada quanto para o de saída.

```
# Matriz de regressao:
nb = 4
na = 10
ni = np.arange(na, Ni + na)
M = np.zeros((Ni, na + nb))

# Para regressores de y:
for l in np.arange(0, na):
    M[:, l] = yout[ni - l - 1]

# Para regressores de u:
for l in np.arange(0, nb):
    M[:, na+l] = u1[ni-l]

print(M.shape)
```

Com as matrizes de regressores ajustada para um sistema de décima ordem, foi realizada a aplicação do método dos mínimos quadrados.

```
# Minimos quadrados
thetaA = np.linalg.inv(M.T @ M) @ M.T @ yout[ni]
thetaA
```

A equação 3.3 possui 4 coeficientes no numerador e 10 no denominador, dessa forma, as quatro primeiras posições da variável **thetaA** corresponde aos coeficientes do numerador e o restante aos coeficientes do denominador, a célula de código abaixo define a função de transferência usando a estrutura da equação 3.3 e os coeficientes obtidos.

```
a1, a2, a3, a4, a5, a6, a7, a8, a9, a10 = thetaA[:10]
b0, b1, b2, b3 = thetaA[10:]

Ba = [b0, b1, b2, b3]
```

```
Aa = [1, -a1, -a2, -a3, -a4, -a5, -a6, -a7, -a8, -a9, -a10]

Gz = ct.tf(Ba, Aa, Ts)
```

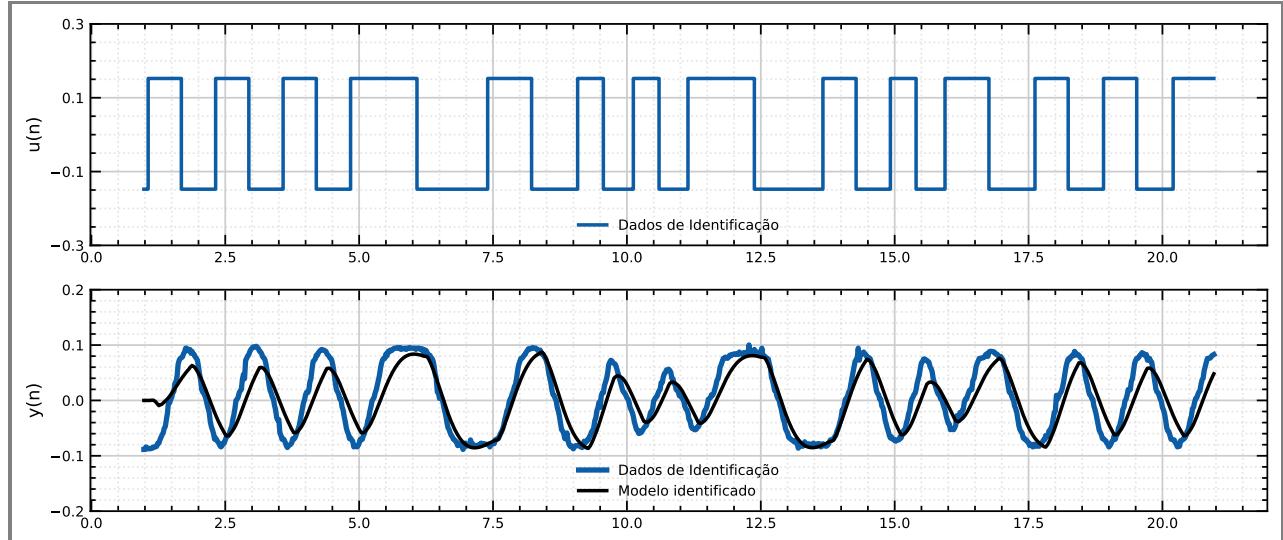
Assim, obtém-se a função de transferência discreta, equação 3.4, com os valores encontrado a partir do método de mínimos quadrados.

$$\frac{-0.0029z^3 + 0.0023z^2 + 0.0016z + 0.0097}{z^{10} - 0.9z^9 - 0.3z^8 - 0.014z^7 + 0.13z^6 + 0.15z^5 + 0.012z^4 - 0.05z^3 - 0.12z^2 - 0.02z + 0.15} \quad dt = 0.02 \quad (3.4)$$

Por fim, realizou-se novamente a simulação do modelo encontrado, com o propósito de compará-lo aos dados de saída obtidos durante o ensaio do sistema real, a célula de código abaixo implementa a simulação.

```
_ , yp = ct.forced_response(Gz, U=u1)
```

Figura 28 – Validação do modelo de segundo grau.



Fonte: elaborado pelo autor (2023).

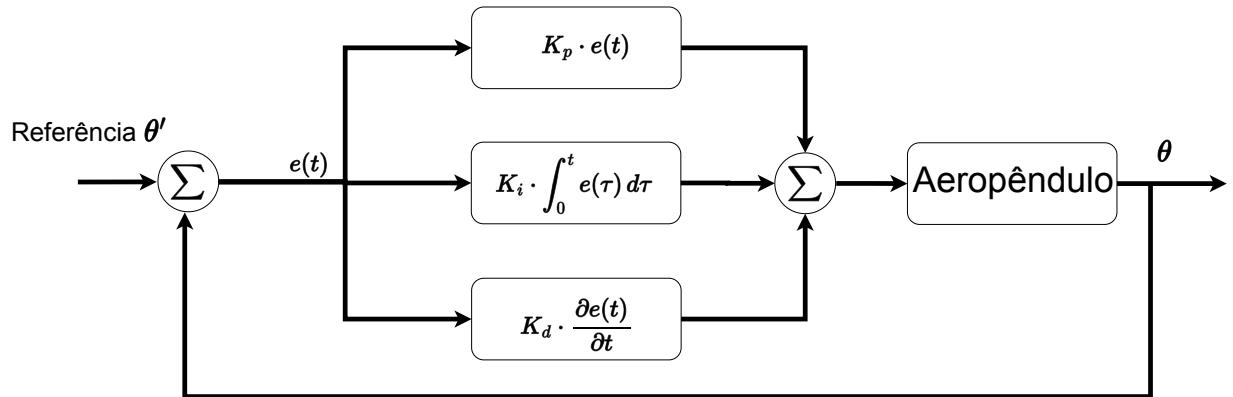
A partir de uma análise qualitativa entre o sinal de saída real e o simulado, figura 28, pode-se concluir que o modelo proposto a partir do método de mínimos quadrados teve sucesso, no entanto, ao modelar sistemas reais sempre haverá erro, isso por conta da não linearidade dos componentes físicos: sensores, atuadores e do próprio sistema, sendo assim, o modelo obtido é apenas uma aproximação do sistema real.

### 3.3 Ensaio em Malha Fechada com Controlador PID

O ecossistema desenvolvido possibilita aplicar diferentes controladores, bastando implementá-lo no formato de uma biblioteca e chamá-la no código principal do firmware, passando como parâmetro o erro que é calculado subtraindo da referência o valor lido na saída do sistema, isso permite maior flexibilidade ao pesquisador, que pode projetar diferentes controladores e realizar seus testes sem que seja necessário sobrescrever o código de implementação do sistema de controle, ou seja, a equação do controlador.

Com propósito de testar o sistema em malha fechada, foi desenvolvido um controlador PID simples, tendo apenas um requisito, erro nulo para o sinal degrau. A Figura 29 exibe o diagrama do controlador proposto.

Figura 29 – Sistema em Malha Fechada com Controlador PID.



Fonte: elaborado pelo autor (2023).

No contexto da validação do sistema em malha fechada, optou-se por uma abordagem simplificada para a sintonia do controlador PID. Este método baseou-se na técnica de tentativa e erro. Após uma série de experimentos, determinaram-se os valores ótimos para os coeficientes do controlador, que foram os seguintes:  $K_p = 0,02$ ,  $K_i = 0,055$  e  $K_d = 0,35$ .

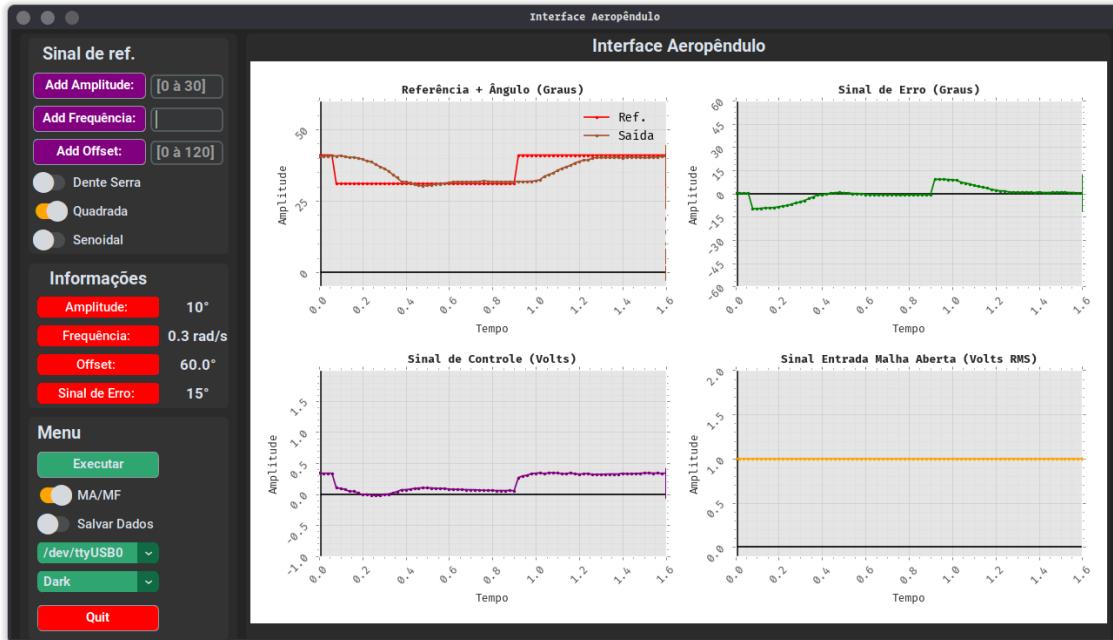
#### 3.3.1 Onda Quadrada

Ao inicializar a interface é possível escolher a opção de malha fechada, com isso a interface envia um comando para o microcontrolador que fecha a malha e aplica o sinal de controle na entrada do sistema, dessa forma, o aeropêndulo rasteia o sinal de referência escolhido, para o primeiro teste foi escolhido como sinal de referência uma onda quadrada com os seguintes parâmetros, frequência de 0,5 Hz, Amplitude 15° e offset de 1V. A figura 30, permite visualizar os sinais de referência e saída, ao analisá-los pode-se observar que o sinal de saída rasteia o sinal de referência. Além disso, observa-se que o sistema possui um transitório nas extremidades do sinal de onda

quadrada, isso pode ser analisado a partir do sinal de erro que aumenta as extremidades do sinal de referência.

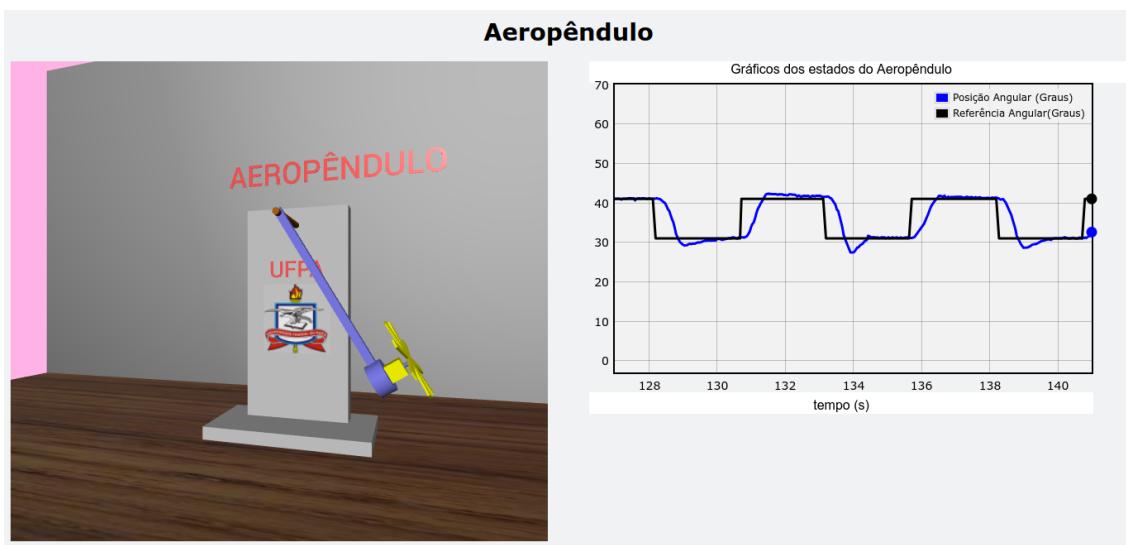
O gêmeo digital consome em tempo real o sinal angular do sistema real capturado através do potenciômetro, com isso é possível visualizar de uma forma gráfica a dinâmica do sistema real.

Figura 30 – Gráficos dos Estados do Aeropêndulo com Controlador PID.



Fonte: elaborado pelo autor (2023).

Figura 31 – Gêmeo Digital com Controlador PID.

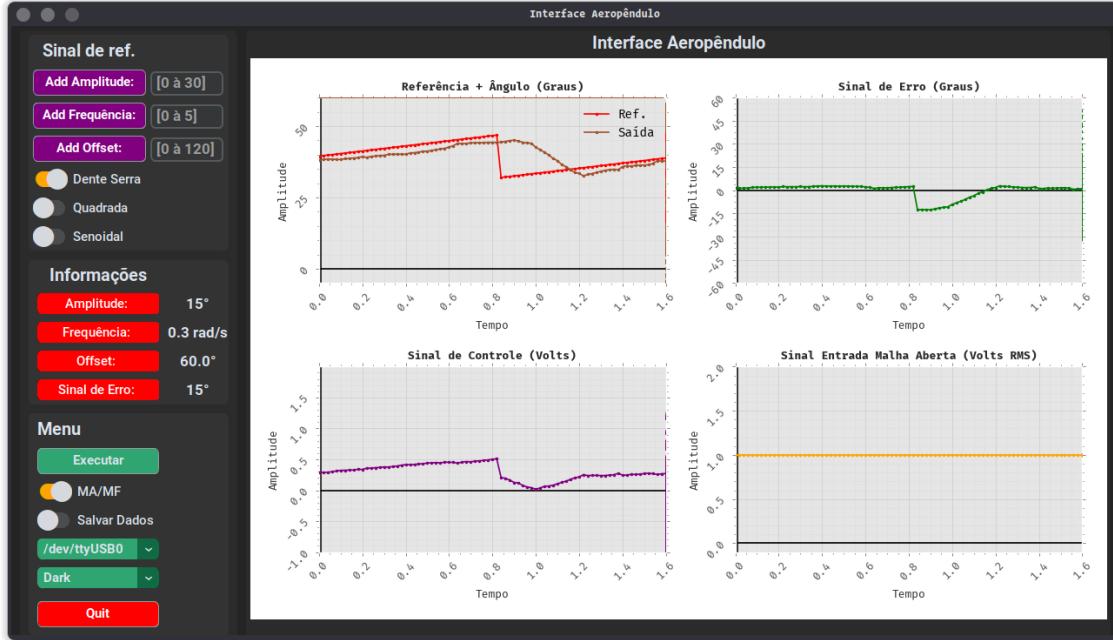


Fonte: elaborado pelo autor (2023).

### 3.3.2 Onda Dente de Serra

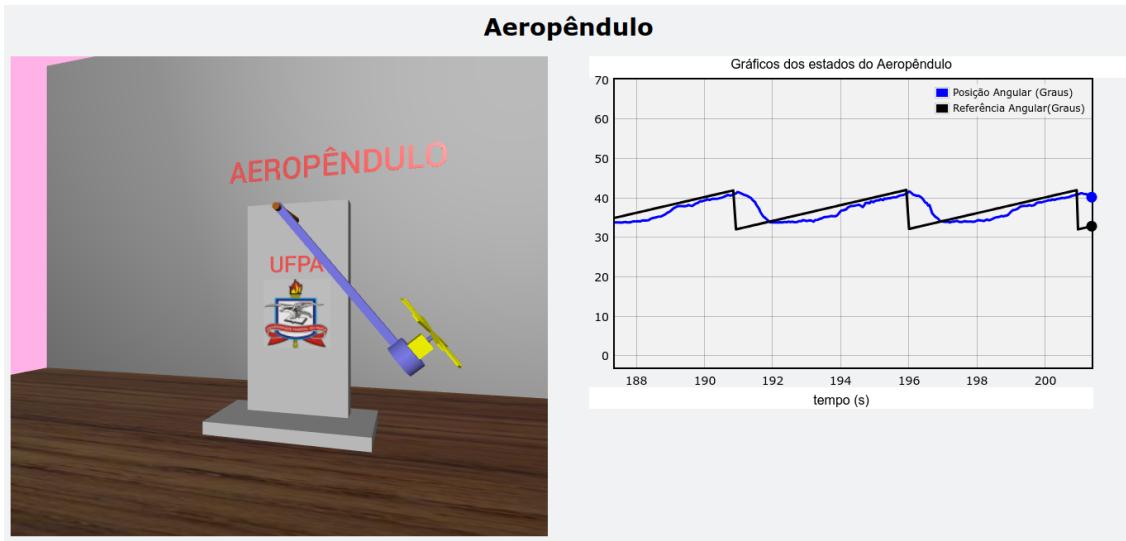
Com o sinal de referência sendo uma onda dente de serra, pode-se observar que o controlador consegue rastreá-lo, tendo assim como no sinal degrau, um transitório nas extremidades.

Figura 32 – Sistema em Malha Fechada com Controlador PID.



Fonte: elaborado pelo autor (2023).

Figura 33 – Gêmeo Digital consumindo os dados em tempo real do protótipo em Malha Fechada com Controlador PID.



Fonte: elaborado pelo autor (2023).



# CONCLUSÃO

## 4.1 Considerações Finais

O ecossistema concebido para este projeto foi meticulosamente desenvolvido e testado, com resultados positivos abrangendo a modelagem matemática do sistema, a concepção e construção do protótipo, a elaboração do firmware, bem como o desenvolvimento dos softwares para a interface com o Aeropêndulo e a criação do gêmeo digital.

A modelagem matemática do sistema foi fundamentada nos princípios de Newton, no entanto, este processo demonstrou que a modelagem de sistemas pode rapidamente se tornar complexa e impraticável. Mesmo após a obtenção de um modelo que descreva a dinâmica do sistema, a determinação dos coeficientes finais pode ser uma tarefa árdua, uma vez que requer a utilização de sensores para obter tais grandezas, acrescentando uma camada de complexidade e desafios ao processo de modelagem. Dessa forma, o método de identificação de sistemas pode ser aplicado para encontrar um modelo matemático que aproxime a dinâmica do sistema real.

A concepção do protótipo foimeticulosamente planejada visando a simplificação do processo de construção. Com esse propósito em mente, a escolha recaiu sobre a utilização de materiais de custo reduzido para a estrutura e componentes eletrônicos amplamente disponíveis. Tal abordagem resultou na capacidade de replicar o sistema de forma eficaz e acessível, ampliando seu alcance não apenas para o público acadêmico, mas também para entusiastas interessados em sua implementação.

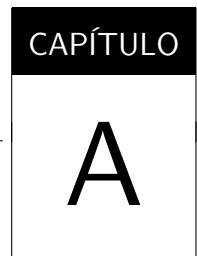
Após a implementação do ecossistema, foram conduzidos testes essenciais para validar o projeto. Inicialmente, adotou-se a técnica de identificação de sistema, que se mostrou eficaz na criação de um modelo preciso com base nos dados de entrada e saída do sistema. Em uma etapa subsequente, aplicamos um controlador PID em malha fechada para avaliar o desempenho do

sistema em condições práticas. Ambos os testes resultaram em êxito, fornecendo evidências sólidas de que os objetivos do projeto foram plenamente alcançados. Com base nesses resultados, o projeto foi validado com sucesso e se encontra pronto para ser utilizado em disciplinas relacionadas a sistemas de controle, bem como para a realização de pesquisas acadêmicas, contribuindo para o avanço do conhecimento nessa área.

## 4.2 Trabalhos Futuros

O projeto está agora concluído e totalmente preparado para ser aplicado em diversas áreas de pesquisa. Essa versatilidade permite a realização de estudos em identificação de sistemas, explorando diferentes métodos. Além disso, é viável o desenvolvimento de controladores por meio de abordagens clássicas ou com o uso de inteligência artificial, incluindo técnicas de aprendizagem por reforço, deep Q-learning e outros algoritmos relevantes. Além disso, a expansão do ecossistema é factível, permitindo a incorporação de novas funcionalidades tanto na interface gráfica quanto no próprio protótipo.

O projeto avançará com a elaboração de uma documentação abrangente, a qual será disponibilizada online através do GitHub, apêndice A. Além disso, o recurso será completado com a criação de vídeos explicativos que detalharão cada aspecto do projeto. É importante destacar que o projeto será de código aberto, aberto a contribuições de pesquisadores, estudantes e entusiastas. Essa abertura colaborativa permitirá um contínuo aprimoramento ao longo do tempo. Dessa forma, nosso objetivo é estabelecer um ambiente de pesquisa abrangente e acessível, promovendo a ampla disseminação do conhecimento na comunidade acadêmica e além dela.



# REPOSITÓRIO NO GITHUB

Link repositório: [https://github.com/Oseiasdfarias/Projeto\\_Tcc\\_Oseias\\_Oficial](https://github.com/Oseiasdfarias/Projeto_Tcc_Oseias_Oficial)

Figura 34 – Repositório no GitHub.

The screenshot shows the GitHub interface for the repository 'Projeto\_Tcc\_Oseias\_Oficial'. The repository is private and has 303 commits. The 'About' section describes the project as 'Aeropêndulo: Implementação de um Ecossistema para Estudos de Controle e Modelagem de Sistemas'. The 'Readme' section lists tags: prototipagem, engenharia-eletrica, sistemas-de-controle, aeropendulo. The 'Activity' section shows 2 watching and 0 forks. The 'Releases' section is currently empty.

Commit	Author	Message	Date	Time
fb798c6	Oseiasdfarias	atualização do repositório com novos arquivos	3 days ago	12:45
303 commits				
3 days ago				
3 days ago				
10 months ago				
4 months ago				
7 months ago				
3 days ago				
3 months ago				
5 months ago				
7 months ago				
3 weeks ago				

Fonte: elaborado pelo autor (2023).

# REFERÊNCIAS

---



---

- COTA, Y. d. O. Apresentação do artigo técnico elaborado em 2023 por meio da in proeg n° 01/2023: explorando o potencial do laboratório virtual de controle de sistemas com vpython no ensino de engenharia: análise e perspectivas para o futuro. 2023. 2
- KLARISSA, S. Identificação não-linear no espaço de estados por regressão esparsa de bancada motor-gerador. *Faculdade de Engenharia elétrica, Universidade Federal do Pará - Campus Tucuruí, Pará, 2023.* 37
- LICEAGA-CASTRO, J. U.; SILLER-ALCALÁ, I. I.; JAIMES-PONCE, J.; ALCÁNTARA-RAMÍREZ, R. Series dc motor modeling and identification. In: *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*. [S.l.: s.n.], 2017. p. 248–253. 10, 11, 12
- LIECHTI CHRIS. *pySerial documentação Oficial*. 2023. Disponível em: <<https://pyserial.readthedocs.io/en/latest/pyserial.html>>. Acesso em: 01 de outubro 2023. 31
- MATPLOTLIB DEVELOPMENT TEAM. *Site oficial da biblioteca Matplotlib*. 2023. Disponível em: <<https://matplotlib.org>>. Acesso em: 01 de outubro 2023. 31
- MOHAMMADBAGHERI, A.; YAGHOOBI, M. A new approach to control a driven pendulum with pid method. In: *2011 UtkSim 13th International Conference on Computer Modelling and Simulation*. [S.l.: s.n.], 2011. p. 207–211. 8
- NETO ANSELMO. *Introdução ao Numerical Python (Numpy)*. 2023. Disponível em: <<http://www.onl.ufc.br/post/numpy>>. Acesso em: 01 de outubro 2023. 32
- NISE, S. N. *Engenharia de sistemas de controle*. [S.l.]: LTC, 2013. v. 3. ISBN 9788521621362. 1
- OGATA, K. *Engenharia de Controle Moderno*. Brasil: Pearson Education do Brasil, 2014. 7
- PANDAS DEVELOPMENT TEAM. *Site oficial da biblioteca Pandas*. 2023. Disponível em: <<https://pandas.pydata.org>>. Acesso em: 01 de outubro 2023. 32
- SCKIMANSKY TOM. *Site oficial da biblioteca CustomTkinter*. 2023. Disponível em: <<https://customtkinter.tomschimansky.com>>. Acesso em: 01 de outubro 2023. 31
- UMANS, S. *Máquinas Elétricas de Fitzgerald e Kingsley - 7.ed.* AMGH Editora, 2014. ISBN 9788580553741. Disponível em: <<https://books.google.com.br/books?id=3Fa2AwAAQBAJ>>. 10
- VPYTHON.ORG. *Site oficial da biblioteca VPython*. 2023. Disponível em: <<https://customtkinter.tomschimansky.com>>. Acesso em: 01 de outubro 2023. 26